Trident 사용

Trident

NetApp
January 15, 2026

목차

Trident 사용

워커 노드 준비

Kubernetes 클러스터의 모든 워커 노드는 Pod에 대해 프로비저닝한 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS, iSCSI, NVMe/TCP 또는 FC 도구를 설치해야 합니다.

올바른 도구 선택

여러 드라이버를 조합하여 사용하는 경우 드라이버에 필요한 모든 도구를 설치해야 합니다. Red Hat Enterprise Linux CoreOS(RHCOS)의 최신 버전에는 기본적으로 도구가 설치되어 있습니다.

NFS 도구

"NFS 도구 설치"다음을 사용하는 경우: `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` .

iSCSI 도구

"iSCSI 도구 설치"다음을 사용하는 경우: `ontap-san` , `ontap-san-economy` , `solidfire-san` .

NVMe 도구

"NVMe 도구 설치"당신이 사용하는 경우 `ontap-san` TCP(NVMe/TCP) 프로토콜을 통한 비휘발성 메모리 익스프레스(NVMe)용입니다.

NetApp NVMe/TCP에 ONTAP 9.12 이상을 권장합니다.

FC 도구를 통한 **SCSI**

참조하다"FC 및 FC-NVMe SAN 호스트를 구성하는 방법" FC 및 FC-NVMe SAN 호스트 구성에 대한 자세한 내용은 다음을 참조하세요.

"FC 도구 설치"당신이 사용하는 경우 `ontap-san` sanType으로 `fcp` (FC를 통한 SCSI).

고려 사항: * FC를 통한 SCSI는 OpenShift 및 KubeVirt 환경에서 지원됩니다. * Docker에서는 FC를 통한 SCSI가 지원되지 않습니다. * iSCSI 자체 복구 기능은 FC를 통한 SCSI에는 적용되지 않습니다.

노드 서비스 검색

Trident 노드가 iSCSI 또는 NFS 서비스를 실행할 수 있는지 자동으로 감지하려고 시도합니다.

노드 서비스 검색은 검색된 서비스를 식별하지만 서비스가 올바르게 구성되었는지 보장하지는 않습니다. 반대로, 검색된 서비스가 없다고 해서 볼륨 마운트가 실패한다는 보장은 없습니다.

이벤트 검토

Trident 노드가 발견된 서비스를 식별할 수 있도록 이벤트를 생성합니다. 이러한 이벤트를 검토하려면 다음을 실행하세요.

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

발견된 서비스 검토

Trident Trident 노드 CR의 각 노드에서 활성화된 서비스를 식별합니다. 검색된 서비스를 보려면 다음을 실행하세요.

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS 볼륨

운영 체제에 맞는 명령을 사용하여 NFS 도구를 설치합니다. 부팅 시 NFS 서비스가 시작되었는지 확인하세요.

RHEL 8 이상

```
sudo yum install -y nfs-utils
```

우분투

```
sudo apt-get install -y nfs-common
```

컨테이너에 볼륨을 연결할 때 오류가 발생하는 것을 방지하려면 NFS 도구를 설치한 후 작업자 노드를 재부팅하세요.

iSCSI 볼륨

Trident 자동으로 iSCSI 세션을 설정하고, LUN을 스캔하고, 다중 경로 장치를 검색하여 포맷하고, 포드에 마운트할 수 있습니다.

iSCSI 자체 복구 기능

ONTAP 시스템의 경우 Trident 다음을 위해 5분마다 iSCSI 자체 복구를 실행합니다.

1. 원하는 iSCSI 세션 상태와 현재 iSCSI 세션 상태를 *식별*합니다.
2. 원하는 상태와 현재 상태를 *비교*하여 필요한 수리를 파악합니다. Trident 수리 우선순위를 결정하고 언제 수리를 먼저 시작해야 할지 결정합니다.
3. 현재 iSCSI 세션 상태를 원하는 iSCSI 세션 상태로 되돌리려면 필요한 수리를 수행합니다.

자체 복구 활동 로그는 다음 위치에 있습니다. `trident-main` 해당 Daemonset 포드의 컨테이너입니다. 로그를 보려면 다음을 설정해야 합니다. `debug` Trident 설치 중에 "true"로 설정합니다.

Trident iSCSI 자체 복구 기능은 다음을 방지하는 데 도움이 될 수 있습니다.

- 네트워크 연결 문제 이후 발생할 수 있는 오래되거나 비정상적 iSCSI 세션입니다. 세션이 오래된 경우, Trident 포털과의 연결을 재설정하기 위해 로그아웃하기 전에 7분을 기다립니다.

> 예를 들어, CHAP 비밀번호가 스토리지 컨트롤러에서 순환되고 네트워크 연결이 끊어지면 이전(*stale*) CHAP 비밀번호가 유지될 수 있습니다. 자가 복구 기능은 이를 인식하고 업데이트된 CHAP 비밀을 적용하기 위해 세션을 자동으로 재설정합니다.

- iSCSI 세션이 누락되었습니다
- LUN이 누락되었습니다
- Trident 업그레이드 전 고려사항*
- 노드당 igroup(23.04+에서 도입)만 사용 중인 경우 iSCSI 자체 복구 기능은 SCSI 버스의 모든 장치에 대한 SCSI 재검색을 시작합니다.
- 백엔드 범위의 igroup(23.04부터 더 이상 사용되지 않음)만 사용 중인 경우 iSCSI 자체 복구 기능은 SCSI 버스에서 정확한 LUN ID를 찾기 위해 SCSI 재검색을 시작합니다.
- 노드별 igroup과 백엔드 범위 igroup을 혼합하여 사용하는 경우 iSCSI 자체 복구 기능은 SCSI 버스에서 정확한 LUN ID를 찾기 위해 SCSI 재검색을 시작합니다.

iSCSI 도구 설치

운영 체제에 맞는 명령을 사용하여 iSCSI 도구를 설치합니다.

시작하기 전에

- Kubernetes 클러스터의 각 노드에는 고유한 IQN이 있어야 합니다. 이것은 필수 전제 조건입니다.
- RHCOS 버전 4.5 이상 또는 기타 RHEL 호환 Linux 배포판을 사용하는 경우 `solidfire-san` 드라이버 및 Element OS 12.5 이하에서는 CHAP 인증 알고리즘이 MD5로 설정되어 있는지 확인하십시오. `/etc/iscsi/iscsid.conf` Element 12.7에서는 보안 FIPS 호환 CHAP 알고리즘 SHA1, SHA-256 및 SHA3-256을 사용할 수 있습니다.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- iSCSI PV와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 다음을 지정합니다. `discard` StorageClass의 mountOption을 사용하여 인라인 공간 회수를 수행합니다. 참조하다 "Red Hat 문서" .
- 최신 버전으로 업그레이드했는지 확인하세요. `multipath-tools` .

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. iscsi-initiator-utils 버전이 6.2.0.874-2.el7 이상인지 확인하세요.

```
rpm -q iscsi-initiator-utils
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `iscsid` 그리고 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now iscsid multipathd
```

6. 활성화하고 시작하세요 `iscsi` :

```
sudo systemctl enable --now iscsi
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic의 경우) 또는 2.0.874-7.1ubuntu6.1 이상(focal의 경우)인지 확인하세요.

```
dpkg -l open-iscsi
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `open-iscsi` 그리고 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Ubuntu 18.04의 경우 대상 포트를 검색해야 합니다. `iscsiadm` 시작하기 전에 `open-iscsi` iSCSI 데몬을 시작하려면. 또는 다음을 수정할 수 있습니다. `iscsi` 서비스를 시작하려면 `iscsid` 자동으로.

iSCSI 자체 복구 구성 또는 비활성화

다음의 Trident iSCSI 자체 복구 설정을 구성하여 오래된 세션을 수정할 수 있습니다.

- **iSCSI** 자체 복구 간격: iSCSI 자체 복구가 호출되는 빈도를 결정합니다(기본값: 5분). 작은 숫자를 설정하면 더 자주 실행되도록 구성할 수 있고, 큰 숫자를 설정하면 덜 자주 실행되도록 구성할 수 있습니다.

> iSCSI 자체 복구 간격을 0으로 설정하면 iSCSI 자체 복구가 완전히 중지됩니다. iSCSI 자체 복구를 비활성화하는 것은 권장하지 않습니다. iSCSI 자체 복구가 의도한 대로 작동하지 않거나 디버깅 목적으로만 특정 상황에서만 비활성화해야 합니다.

- **iSCSI** 자체 복구 대기 시간: iSCSI 자체 복구가 비정상 세션에서 로그아웃하고 다시 로그인을 시도하기 전까지 기다리는 시간을 결정합니다(기본값: 7분). 세션이 비정상적으로 식별되어 로그아웃되기 전에 더 오래 기다린 후 다시 로그인을 시도하도록 하려면 숫자를 더 크게 구성하거나, 로그아웃한 후 더 일찍 로그인하도록 숫자를 더 작게 구성할 수 있습니다.

지배

iSCSI 자체 복구 설정을 구성하거나 변경하려면 다음을 전달하세요. `iscsiSelfHealingInterval` 그리고 `iscsiSelfHealingWaitTime` Helm 설치 또는 Helm 업데이트 중의 매개변수.

다음 예에서는 iSCSI 자체 복구 간격을 3분으로 설정하고 자체 복구 대기 시간을 6분으로 설정합니다.

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

트라이던트**ctl**

iSCSI 자체 복구 설정을 구성하거나 변경하려면 다음을 전달하세요. `iscsi-self-healing-interval` 그리고 `iscsi-self-healing-wait-time` tridentctl 설치 또는 업데이트 중의 매개변수.

다음 예에서는 iSCSI 자체 복구 간격을 3분으로 설정하고 자체 복구 대기 시간을 6분으로 설정합니다.

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCP 볼륨

운영 체제에 맞는 명령을 사용하여 NVMe 도구를 설치하세요.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 커널 버전에서 NVMe 패키지를 사용할 수 없는 경우, 노드의 커널 버전을 NVMe 패키지가 있는 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

우분투

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

설치 확인

설치 후 다음 명령을 사용하여 Kubernetes 클러스터의 각 노드에 고유한 NQN이 있는지 확인하세요.

```
cat /etc/nvme/hostnqn
```

Trident 다음을 수정합니다. `ctrl_device_tmo` NVMe가 다운되더라도 경로를 포기하지 않도록 보장하는 가치입니다. 이 설정을 변경하지 마세요.

FC 볼륨을 통한 SCSI

이제 Trident 와 함께 Fibre Channel(FC) 프로토콜을 사용하여 ONTAP 시스템에서 스토리지 리소스를 프로비저닝하고 관리할 수 있습니다.

필수 조건

FC에 필요한 네트워크 및 노드 설정을 구성합니다.

네트워크 설정

1. 대상 인터페이스의 WWPN을 가져옵니다. 참조하다 "네트워크 인터페이스 표시" 자세한 내용은.
2. 개시자(호스트)의 인터페이스에 대한 WWPN을 가져옵니다.

   해당 호스트 운영 체제 유틸리티를 참조하세요.
3. 호스트와 대상의 WWPN을 사용하여 FC 스위치에서 구역화를 구성합니다.

   자세한 내용은 해당 스위치 공급업체의 설명서를 참조하세요.

   자세한 내용은 다음 ONTAP 문서를 참조하세요.

   - "파이버 채널 및 FCoE 구역화 개요"

- "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"

FC 도구 설치

운영 체제에 맞는 명령을 사용하여 FC 도구를 설치하세요.

- FC PV와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 다음을 지정합니다. `discard` StorageClass의 mountOption을 사용하여 인라인 공간 회수를 수행합니다. 참조하다 "Red Hat 문서" .

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now multipathd
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
```

백엔드 구성 및 관리

백엔드 구성

백엔드는 Trident 와 스토리지 시스템 간의 관계를 정의합니다. 이는 Trident 해당 스토리지 시스템과 통신하는 방법과 Trident 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

Trident 스토리지 클래스에서 정의한 요구 사항에 맞는 백엔드의 스토리지 풀을 자동으로 제공합니다. 스토리지 시스템의 백엔드를 구성하는 방법을 알아보세요.

- "Azure NetApp Files 백엔드 구성"
- "Google Cloud NetApp Volumes 백엔드 구성"
- "Google Cloud Platform 백엔드에 Cloud Volumes Service 구성"
- "NetApp HCI 또는 SolidFire 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버를 사용하여 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버를 사용하여 백엔드 구성"
- "Amazon FSx for NetApp ONTAP 과 함께 Trident 사용"

Azure NetApp Files

Azure NetApp Files 백엔드 구성

Azure NetApp Files Trident 의 백엔드로 구성할 수 있습니다. Azure NetApp Files 백엔드를 사용하여 NFS 및 SMB 볼륨을 연결할 수 있습니다. Trident 또한 Azure Kubernetes Services(AKS) 클러스터에 대한 관리형 ID를 사용하여 자격 증명 관리를 지원합니다.

Azure NetApp Files 드라이버 세부 정보

Trident 클러스터와 통신하기 위해 다음과 같은 Azure NetApp Files 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`azure-netapp-files`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

고려 사항

- Azure NetApp Files 서비스는 50GiB보다 작은 볼륨을 지원하지 않습니다. 더 작은 볼륨이 요청되면 Trident 자동으로 50GiB 볼륨을 생성합니다.
- Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.

AKS의 관리 ID

Trident 지지대"관리되는 ID" Azure Kubernetes Services 클러스터용. 관리형 ID가 제공하는 간소화된 자격 증명 관리를 활용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 관리 ID
- 다음을 포함하는 Trident 설치됨 `cloudProvider` 지정하다 `"Azure"` .

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 `"Azure"` . 예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

지배

다음 예제에서는 Trident 세트를 설치합니다. `cloudProvider` 환경 변수를 사용하여 Azure에 `$CP` :

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>트라이던트ctl</code>

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 플래그를 `Azure` :

```
tridentctl install --cloud-provider="Azure" -n trident
```

AKS용 클라우드 ID

클라우드 ID를 사용하면 Kubernetes Pod가 명시적인 Azure 자격 증명을 제공하는 대신 워크로드 ID로 인증하여 Azure 리소스에 액세스할 수 있습니다.

Azure에서 클라우드 ID를 활용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터

- AKS Kubernetes 클러스터에 구성된 워크로드 ID 및 oidc-issuer
- 다음을 포함하는 Trident 설치됨 `cloudProvider` 지정하다 `"Azure"` 그리고 `cloudIdentity` 워크로드 ID 지정

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 `"Azure"` 그리고 설정하다 `cloudIdentity` 에게 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` .

예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

지배

다음 환경 변수를 사용하여 **cloud-provider (CP)** 및 **cloud-identity (CI)** 플래그의 값을 설정합니다.

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 환경 변수를 사용하여 Azure에 `$CP` 그리고 설정합니다 `cloudIdentity` 환경 변수 사용 `$CI` :

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>트라이던트ctl</code>

다음 환경 변수를 사용하여 클라우드 공급자 및 클라우드 **ID** 플래그의 값을 설정합니다.

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloud-provider` 플래그를 `$CP` , 그리고 `cloud-identity` 에게 `$CI` :

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Azure NetApp Files 백엔드 구성을 준비합니다.

Azure NetApp Files 백엔드를 구성하기 전에 다음 요구 사항이 충족되는지 확인해야 합니다.

NFS 및 **SMB** 볼륨에 대한 필수 구성 요소

Azure NetApp Files 처음 사용하거나 새로운 위치에서 사용하는 경우 Azure NetApp Files를 설정하고 NFS 볼륨을 생성하기 위해 일부 초기 구성이 필요합니다. 참조하다 "Azure: Azure NetApp Files 설정 및 NFS 볼륨 생성" .

구성하고 사용하려면 "Azure NetApp Files" 백엔드에는 다음이 필요합니다.

- `subscriptionID`, `tenantID` , `clientID` , `location` , 그리고 `clientSecret` AKS 클러스터에서 관리형 ID를 사용하는 경우 선택 사항입니다.
- `tenantID`, `clientID` , 그리고 `clientSecret` AKS 클러스터에서 클라우드 ID를 사용하는 경우 선택 사항입니다.

- 수용 인원 풀. 참조하다"Microsoft: Azure NetApp Files 대한 용량 풀 만들기" .
- Azure NetApp Files 에 위임된 서브넷입니다. 참조하다"Microsoft: Azure NetApp Files 에 서브넷 위임" .
- `subscriptionID`Azure NetApp Files 활성화된 Azure 구독에서.
- `tenantID`, `clientID` , 그리고 `clientSecret` 에서"앱 등록" Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory에 있습니다. 앱 등록에는 다음 중 하나를 사용해야 합니다.
  - 소유자 또는 기여자 역할"Azure에서 미리 정의됨" .
  - 에이"사용자 정의 기여자 역할" 구독 수준에서(`assignableScopes`) Trident 에 필요한 것으로만 제한된 다음 권한이 있습니다. 사용자 정의 역할을 만든 후,"Azure Portal을 사용하여 역할 할당" .

사용자 정의 기여자 역할

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/delete",
          "Microsoft.Features/features/read",
          "Microsoft.Features/operations/read",
          "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

- Azure `location` 적어도 하나 이상 포함 "위임된 서브넷" . Trident 22.01부터 `location` 매개변수는 백엔드 구성 파일의 최상위에 있는 필수 필드입니다. 가상 풀에 지정된 위치 값은 무시됩니다.
- 사용하려면 `Cloud Identity` , 얻으세요 `client ID` 에서 "사용자가 할당한 관리 ID" 그리고 해당 ID를 지정하세요 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` .

SMB 볼륨에 대한 추가 요구 사항

SMB 볼륨을 생성하려면 다음이 필요합니다.

- Active Directory가 구성되고 Azure NetApp Files 에 연결되었습니다. 참조하다"Microsoft: Azure NetApp Files 대한 Active Directory 연결 만들기 및 관리" .
- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Azure NetApp Files Active Directory에 인증할 수 있도록 Active Directory 자격 증명이 포함된 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

Azure NetApp Files 백엔드 구성 옵션 및 예제

Azure NetApp Files 에 대한 NFS 및 SMB 백엔드 구성 옵션에 대해 알아보고 구성 예를

검토하세요.

백엔드 구성 옵션

Trident 백엔드 구성(서브넷, 가상 네트워크, 서비스 수준 및 위치)을 사용하여 요청된 위치에서 사용 가능하고 요청된 서비스 수준 및 서브넷과 일치하는 용량 풀에 Azure NetApp Files 볼륨을 만듭니다.

* NetApp Trident 25.06 릴리스부터 수동 QoS 용량 풀이 기술 미리 보기로 지원됩니다.*

Azure NetApp Files 백엔드는 다음과 같은 구성 옵션을 제공합니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	"azure-netapp-파일"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	운전자 이름 + "_" + 임의의 문자
`subscriptionID`	AKS 클러스터에서 관리 ID가 활성화된 경우 Azure 구독의 구독 ID는 선택 사항입니다.	
`tenantID`	AKS 클러스터에서 관리 ID 또는 클라우드 ID가 사용되는 경우 앱 등록 선택 사항의 테넌트 ID입니다.	
`clientID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID가 사용되는 경우 앱 등록 선택 사항의 클라이언트 ID입니다.	
`clientSecret`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록 선택 사항의 클라이언트 비밀번호입니다.	
`serviceLevel`	중 하나 `Standard` , `Premium` , 또는 `Ultra`	"" (무작위의)
`location`	새 볼륨이 생성될 Azure 위치의 이름입니다. AKS 클러스터에서 관리 ID가 활성화된 경우 선택 사항입니다.	
`resourceGroups`	검색된 리소스를 필터링하기 위한 리소스 그룹 목록	"[]" (필터 없음)
`netappAccounts`	검색된 리소스를 필터링하기 위한 NetApp 계정 목록	"[]" (필터 없음)
`capacityPools`	검색된 리소스를 필터링하기 위한 용량 풀 목록	"[]" (필터 없음, 무작위)
`virtualNetwork`	위임된 서브넷이 있는 가상 네트워크의 이름	""
`subnet`	위임된 서브넷의 이름 `Microsoft.Netapp/volumes`	""

매개변수	설명	기본
`networkFeatures`	볼륨에 대한 VNet 기능 세트는 다음과 같습니다. `Basic` 또는 `Standard` . 네트워크 기능은 일부 지역에서는 제공되지 않으며 구독을 통해 활성화해야 할 수도 있습니다. 지정 `networkFeatures` 해당 기능이 활성화되지 않으면 볼륨 프로비저닝이 실패합니다.	""
`nfsMountOptions`	NFS 마운트 옵션에 대한 세부적인 제어. SMB 볼륨에서는 무시됩니다. NFS 버전 4.1을 사용하여 볼륨을 마운트하려면 다음을 포함합니다. `nfsvers=4` 쉼표로 구분된 마운트 옵션 목록에서 NFS v4.1을 선택하세요. 스토리지 클래스 정의에 설정된 마운트 옵션은 백엔드 구성에 설정된 마운트 옵션보다 우선합니다.	"nfsvers=3"
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, `\{"api": false, "method": true, "discovery": true}` . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 `nfs` , `smb` 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .	
`qosType`	QoS 유형(자동 또는 수동)을 나타냅니다. * Trident 25.06 기술 미리보기*	자동
`maxThroughput`	허용되는 최대 처리량을 MiB/초 단위로 설정합니다. 수동 QoS 용량 풀에서만 지원됩니다. * Trident 25.06 기술 미리보기*	`4 MiB/sec`

네트워크 기능에 대한 자세한 내용은 다음을 참조하세요."Azure NetApp Files 볼륨에 대한 네트워크 기능 구성" .

필요한 권한 및 리소스

PVC를 생성할 때 "용량 풀을 찾을 수 없습니다" 오류가 발생하는 경우 앱 등록에 필요한 권한과 리소스(서브넷, 가상 네트워크, 용량 풀)가 연결되어 있지 않을 가능성이 높습니다. 디버그가 활성화된 경우 Trident 백엔드가 생성될 때 검색된 Azure 리소스를 기록합니다. 적절한 역할이 사용되고 있는지 확인하세요.

에 대한 값 `resourceGroups` , `netappAccounts` , `capacityPools` , `virtualNetwork` , 그리고 `subnet` 짧은 이름이나 완전한 이름을 사용하여 지정할 수 있습니다. 대부분의 상황에서는 완전한 이름을 사용하는 것이 좋습니다. 짧은 이름은 동일한 이름을 가진 여러 리소스와 일치할 수 있기 때문입니다.

그만큼 `resourceGroups` , `netappAccounts` , 그리고 `capacityPools` 값은 검색된 리소스 세트를 이 스토리지 백엔드에서 사용 가능한 리소스로 제한하는 필터이며, 원하는 대로 조합하여 지정할 수 있습니다. 완전히 정의된 이름은 다음 형식을 따릅니다.

유형	체재
리소스 그룹	<리소스 그룹>
NetApp 계정	<리소스 그룹>/<netapp 계정>
용량 풀	<리소스 그룹>/<netapp 계정>/<용량 풀>
가상 네트워크	<리소스 그룹>/<가상 네트워크>
서브넷	<리소스 그룹>/<가상 네트워크>/<서브넷>

볼륨 프로비저닝

구성 파일의 특정 섹션에서 다음 옵션을 지정하여 기본 볼륨 프로비저닝을 제어할 수 있습니다. 참조하다 구성 예 자세한 내용은.

매개변수	설명	기본
`exportRule`	새로운 볼륨에 대한 내보내기 규칙. `exportRule` CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷의 조합을 쉼표로 구분한 목록이어야 합니다. SMB 볼륨에서는 무시됩니다.	"0.0.0.0/0"
`snapshotDir`	.snapshot 디렉토리의 가시성을 제어합니다.	NFSv4의 경우 "true", NFSv3의 경우 "false"
`size`	새 볼륨의 기본 크기	"100G"
`unixPermissions`	새로운 볼륨의 유닉스 권한(8진수 4자리). SMB 볼륨에서는 무시됩니다.	""(미리보기 기능, 구독 시 허용 목록에 추가 필요)

구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 백엔드의 최소 구성입니다. 이 구성을 사용하면 Trident 구성된 위치에서 Azure NetApp Files 에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀 및 서브넷 중 하나에 무작위로 새 볼륨을 배치합니다. 왜냐하면 `nasType` 생략됩니다. `nfs` 기본값이 적용되고 백엔드가 NFS 볼륨을 프로비저닝합니다.

이 구성은 Azure NetApp Files 처음 시작하고 여러 가지를 시도해 볼 때 이상적이지만 실제로는 프로비저닝하는 볼륨에 대한 추가 범위를 제공해야 합니다.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

AKS의 관리 ID

이 백엔드 구성에서는 다음을 생략합니다. `subscriptionID` , `tenantID` , `clientID` , 그리고 `clientSecret` 관리형 ID를 사용할 때는 선택 사항입니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

AKS용 클라우드 ID

이 백엔드 구성에서는 다음을 생략합니다. `tenantID` , `clientID` , 그리고 `clientSecret` 클라우드 ID를 사용할 때는 선택 사항입니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

용량 풀 필터를 사용한 특정 서비스 수준 구성

이 백엔드 구성은 Azure의 볼륨을 배치합니다. `eastus` 위치 `Ultra` 용량 풀. Trident 해당 위치에서 Azure NetApp Files 에 위임된 모든 서브넷을 자동으로 검색하고 그 중 하나에 무작위로 새 볼륨을 배치합니다.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

수동 **QoS** 용량 풀을 사용한 백엔드 예제

이 백엔드 구성은 Azure의 볼륨을 배치합니다. `eastus` 수동 QoS 용량 풀이 있는 위치입니다. * NetApp Trident 25.06의 기술 미리보기*.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

고급 구성

이 백엔드 구성은 볼륨 배치 범위를 단일 서브넷으로 더욱 줄이고 일부 볼륨 프로비저닝 기본값도 수정합니다.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 개의 스토리지 풀을 정의합니다. 이 기능은 다양한 서비스 수준을 지원하는 여러 용량 풀이 있고 이를 나타내는 Kubernetes의 스토리지 클래스를 생성하려는 경우에 유용합니다. 가상 풀 레이블은 다음을 기준으로 풀을 구별하는 데 사용되었습니다. `performance` .

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - ultra-1
      - ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - standard-1
      - standard-2
```

지원되는 토폴로지 구성

Trident 지역 및 가용성 영역에 따라 워크로드에 대한 볼륨 프로비저닝을 용이하게 합니다. 그만큼 `supportedTopologies` 이 백엔드 구성의 블록은 백엔드당 지역 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 지역 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 지역 및 영역 값과 일치해야 합니다. 이러한 지역과 영역은 저장 클래스에서 제공될 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공하는 지역 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우, Trident 언급된 지역 및 영역에 볼륨을 생성합니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

스토리지 클래스 정의

다음 `StorageClass` 정의는 위의 저장 풀을 참조합니다.

다음을 사용한 정의 예 `parameter.selector` 필드

사용 중 `parameter.selector` 각각에 대해 지정할 수 있습니다 `StorageClass` 볼륨을 호스팅하는 데 사용되는 가상 풀입니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

SMB 볼륨에 대한 예제 정의

사용 중 `nasType` , `node-stage-secret-name` , 그리고 `node-stage-secret-namespace` SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`SMB 볼륨을 지원하는 풀에 대한 필터입니다. `nasType: nfs 또는 nasType: null NFS 풀에 대한 필터.

백엔드 만들기

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes 백엔드 구성

이제 Google Cloud NetApp Volumes Trident 의 백엔드로 구성할 수 있습니다. Google Cloud NetApp Volumes 백엔드를 사용하여 NFS 및 SMB 볼륨을 연결할 수 있습니다.

Google Cloud NetApp Volumes 드라이버 세부 정보

Trident 다음을 제공합니다. `google-cloud-netapp-volumes` 클러스터와 통신하는 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`google-cloud-netapp-volumes`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

GKE용 클라우드 ID

클라우드 ID를 사용하면 Kubernetes 포드가 명시적인 Google Cloud 자격 증명을 제공하는 대신 워크로드 ID로 인증하여 Google Cloud 리소스에 액세스할 수 있습니다.

Google Cloud에서 클라우드 ID를 활용하려면 다음이 필요합니다.

- GKE를 사용하여 배포된 Kubernetes 클러스터.
- GKE 클러스터에 구성된 워크로드 ID와 노드 풀에 구성된 GKE 메타데이터 서버.
- Google Cloud NetApp Volumes 관리자(roles/netapp.admin) 역할 또는 사용자 지정 역할이 있는 GCP 서비스 계정.
- "GCP"를 지정하는 cloudProvider와 새로운 GCP 서비스 계정을 지정하는 cloudIdentity를 포함하는 Trident

설치되었습니다. 아래에 예를 들어보겠습니다.

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 "GCP" 그리고 설정하다 `cloudIdentity` 에게 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com` .

예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

지배

다음 환경 변수를 사용하여 **cloud-provider (CP)** 및 **cloud-identity (CI)** 플래그의 값을 설정합니다.

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 환경 변수를 사용하여 GCP에 `$CP` 그리고 설정합니다 `cloudIdentity` 환경 변수 사용 `$ANNOTATION` :

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>트라이던트ctl</code>

다음 환경 변수를 사용하여 클라우드 공급자 및 클라우드 **ID** 플래그의 값을 설정합니다.

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloud-provider` 플래그를 `$CP` , 그리고 `cloud-identity` 에게 `$ANNOTATION` :

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

Google Cloud NetApp Volumes 백엔드 구성을 준비하세요

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 요구 사항이 충족되는지 확인해야 합니다.

NFS 볼륨의 전제 조건

Google Cloud NetApp Volumes 처음 사용하거나 새로운 위치에서 사용하는 경우 Google Cloud NetApp Volumes 설정하고 NFS 볼륨을 생성하기 위해 초기 구성이 필요합니다. 참조하다"시작하기 전에" .

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 사항이 있는지 확인하세요.

- Google Cloud NetApp Volumes 서비스로 구성된 Google Cloud 계정입니다. 참조하다"Google Cloud NetApp Volumes" .
- Google Cloud 계정의 프로젝트 번호입니다. 참조하다"프로젝트 식별" .
- NetApp Volumes Admin이 있는 Google Cloud 서비스 계정(`roles/netapp.admin`) 역할. 참조하다"ID 및 액세스 관리 역할 및 권한" .
- GCNV 계정의 API 키 파일입니다. 참조하다"서비스 계정 키 생성"
- 저장 풀. 참조하다"스토리지 풀 개요" .

Google Cloud NetApp Volumes 에 대한 액세스를 설정하는 방법에 대한 자세한 내용은 다음을 참조하세요."Google Cloud NetApp Volumes 에 대한 액세스 설정" .

Google Cloud NetApp Volumes 백엔드 구성 옵션 및 예시

Google Cloud NetApp Volumes 의 백엔드 구성 옵션에 대해 알아보고 구성 예를 검토하세요.

백엔드 구성 옵션

각 백엔드는 단일 Google Cloud 지역에서 볼륨을 프로비저닝합니다. 다른 지역에 볼륨을 생성하려면 추가 백엔드를 정의할 수 있습니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	의 가치 `storageDriverName` "google-cloud-netapp-volumes"로 지정해야 합니다.
`backendName`	(선택 사항) 스토리지 백엔드의 사용자 정의 이름	드라이버 이름 + "_" + API 키의 일부

매개변수	설명	기본
`storagePools`	볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호. 해당 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	
`location`	Trident GCNV 볼륨을 생성하는 Google Cloud 위치입니다. 지역 간 Kubernetes 클러스터를 생성할 때 볼륨이 생성됩니다. `location` 여러 Google Cloud 지역의 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.	
`apiKey`	Google Cloud 서비스 계정에 대한 API 키 `netapp.admin` 역할. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다 (백엔드 구성 파일에 그대로 복사됨). 그만큼 `apiKey` 다음 키에 대한 키-값 쌍을 포함해야 합니다. `type` , `project_id` , `client_email` , `client_id` , `auth_uri` , `token_uri` , `auth_provider_x509_cert_url` , 그리고 `client_x509_cert_url` .	
`nfsMountOptions`	NFS 마운트 옵션에 대한 세부적인 제어.	"nfsvers=3"
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`serviceLevel`	스토리지 풀의 서비스 수준과 볼륨입니다. 값은 다음과 같습니다 `flex` , `standard` , `premium` , 또는 `extreme` .	
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`network`	GCNV 볼륨에 Google Cloud 네트워크가 사용됩니다.	
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, `{"api":false, "method":true}` . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 `nfs` , `smb` 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" . 예를 들어: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

볼륨 프로비저닝 옵션

기본 볼륨 프로비저닝을 제어할 수 있습니다. `defaults` 구성 파일의 섹션.

매개변수	설명	기본
`exportRule`	새로운 볼륨에 대한 내보내기 규칙. IPv4 주소의 조합을 쉼표로 구분하여 나열해야 합니다.	"0.0.0.0/0"
`snapshotDir`	에 대한 액세스 `.snapshot` 예배 규칙서	NFSv4의 경우 "true", NFSv3의 경우 "false"
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 백분율	"" (기본값 0 허용)
`unixPermissions`	새로운 볼륨의 유닉스 권한(8진수 4자리).	""

구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 백엔드의 최소 구성입니다. 이 구성을 사용하면 Trident 구성된 위치에서 Google Cloud NetApp Volumes에 위임된 모든 스토리지 풀을 검색하고 해당 풀 중 하나에 무작위로 새 볼륨을 배치합니다. 왜냐하면 `nasType` 생략됩니다. `nfs` 기본값이 적용되고 백엔드가 NFS 볼륨을 프로비저닝합니다.

이 구성은 Google Cloud NetApp Volumes 처음 시작하고 여러 가지를 시도해 볼 때 이상적이지만 실제로는 프로비저닝하는 볼륨에 대한 추가 범위를 제공해야 할 가능성이 높습니다.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

SMB 볼륨 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

StoragePools 필터를 사용한 구성

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 개의 가상 풀을 정의합니다. 가상 풀은 다음에 정의됩니다. `storage` 부분. 여러 개의 스토리지 풀이 서로 다른 서비스 수준을 지원하고 이를 나타내는 Kubernetes의 스토리지 클래스를 만들려는 경우에 유용합니다. 가상 풀 레이블은 풀을 구별하는 데 사용됩니다. 예를 들어, 아래 예에서 `performance` 라벨 및 `serviceLevel` 유형은 가상 풀을 구별하는 데 사용됩니다.

모든 가상 풀에 적용할 수 있는 기본값을 설정하고, 개별 가상 풀의 기본값을 덮어쓸 수도 있습니다. 다음 예에서, `snapshotReserve` 그리고 `exportRule` 모든 가상 풀의 기본값으로 사용됩니다.

자세한 내용은 다음을 참조하세요."가상 풀" .

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
```

```
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKE용 클라우드 ID

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

지원되는 토폴로지 구성

Trident 지역 및 가용성 영역에 따라 워크로드에 대한 볼륨 프로비저닝을 용이하게 합니다. 그만큼 `supportedTopologies` 이 백엔드 구성의 블록은 백엔드당 지역 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 지역 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 지역 및 영역 값과 일치해야 합니다. 이러한 지역과 영역은 저장 클래스에서 제공될 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공하는 지역 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우, Trident 언급된 지역 및 영역에 볼륨을 생성합니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

다음은 무엇인가요?

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
kubectl create -f <backend-file>
```

백엔드가 성공적으로 생성되었는지 확인하려면 다음 명령을 실행하세요.

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 백엔드를 사용하여 설명할 수 있습니다. `kubectl get tridentbackendconfig <backend-name>` 다음 명령을 실행하여 원인을 파악하려면 명령을 실행하거나 로그를 확인하세요.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후 백엔드를 삭제하고 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 정의

다음은 기본입니다 `StorageClass` 위의 백엔드를 참조하는 정의입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

- 다음을 사용한 정의 예 `parameter.selector` 필드:*

사용 중 `parameter.selector` 각각에 대해 지정할 수 있습니다 `StorageClass` 그만큼"가상 풀" 볼륨을 호스팅하는 데 사용됩니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

저장 클래스에 대한 자세한 내용은 다음을 참조하세요."스토리지 클래스 생성" .

SMB 볼륨에 대한 예제 정의

사용 중 `nasType` , `node-stage-secret-name` , 그리고 `node-stage-secret-namespace` SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다. 노드 단계 비밀에는 권한이 있거나 없는 모든 Active Directory 사용자/비밀번호를 사용할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`SMB 볼륨을 지원하는 풀에 대한 필터입니다. `nasType: nfs 또는 nasType: null NFS 풀에 대한 필터.

PVC 정의 예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

PVC가 바인딩되었는지 확인하려면 다음 명령을 실행하세요.

```
kubectl get pvc gcnv-nfs-pvc

NAME           STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc   Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213   100Gi
RWX            gcnv-nfs-sc  1m
```

Google Cloud 백엔드에 대한 Cloud Volumes Service 구성

제공된 샘플 구성을 사용하여 Trident 설치의 백엔드로 Google Cloud용 NetApp Cloud Volumes Service 구성하는 방법을 알아보세요.

Google Cloud 드라이버 세부 정보

Trident 다음을 제공합니다. `gcp-cvs` 클러스터와 통신하는 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`gcp-cvs`	NFS	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`

Google Cloud용 Cloud Volumes Service 에 대한 Trident 지원에 대해 알아보세요.

Trident 두 가지 중 하나에서 Cloud Volumes Service 볼륨을 생성할 수 있습니다."서비스 유형" :

- **CVS-Performance**: 기본 Trident 서비스 유형입니다. 이러한 성능 최적화된 서비스 유형은 성능을 중시하는 프로덕션 워크로드에 가장 적합합니다. CVS-Performance 서비스 유형은 최소 100GiB 크기의 볼륨을 지원하는 하드웨어 옵션입니다. 다음 중 하나를 선택할 수 있습니다."세 가지 서비스 수준" :
  - `standard`
  - `premium`
  - `extreme`
- **CVS**: CVS 서비스 유형은 제한적에서 중간 수준의 성능 수준으로 높은 지역적 가용성을 제공합니다. CVS 서비스 유형은 스토리지 풀을 사용하여 최소 1GiB의 볼륨을 지원하는 소프트웨어 옵션입니다. 스토리지 풀은 최대 50개의 볼륨을 포함할 수 있으며, 모든 볼륨은 풀의 용량과 성능을 공유합니다. 다음 중 하나를 선택할 수 있습니다."두 가지 서비스 수준" :
  - `standardsw`
  - `zoneredundantstandardsw`

필요한 것

구성하고 사용하려면 "Google Cloud용 Cloud Volumes Service" 백엔드에는 다음이 필요합니다.

- NetApp Cloud Volumes Service 로 구성된 Google Cloud 계정
- Google Cloud 계정의 프로젝트 번호
- Google Cloud 서비스 계정 `netappcloudvolumes.admin` 역할
- Cloud Volumes Service 계정의 API 키 파일

백엔드 구성 옵션

각 백엔드는 단일 Google Cloud 지역에서 볼륨을 프로비저닝합니다. 다른 지역에 볼륨을 생성하려면 추가 백엔드를 정의할 수 있습니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	"gcp-cvs"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + API 키의 일부
`storageClass`	CVS 서비스 유형을 지정하는 데 사용되는 선택적 매개변수입니다. 사용 `software` CVS 서비스 유형을 선택하세요. 그렇지 않으면 Trident CVS-Performance 서비스 유형을 가정합니다.(`hardware`).	
`storagePools`	CVS 서비스 유형만 해당. 볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호. 해당 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	
`hostProjectNumber`	공유 VPC 네트워크를 사용하는 경우 필요합니다. 이 시나리오에서는 `projectNumber` 서비스 프로젝트이고, `hostProjectNumber` 호스트 프로젝트입니다.	

매개변수	설명	기본
apiRegion	Trident Cloud Volumes Service 볼륨을 생성하는 Google Cloud 지역입니다. 지역 간 Kubernetes 클러스터를 생성할 때 볼륨이 생성됩니다. apiRegion 여러 Google Cloud 지역의 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.	
apiKey	Google Cloud 서비스 계정에 대한 API 키 netappcloudvolumes.admin 역할. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다(백엔드 구성 파일에 그대로 복사됨).	
proxyURL	CVS 계정에 연결하는 데 프록시 서버가 필요한 경우 프록시 URL입니다. 프록시 서버는 HTTP 프록시 또는 HTTPS 프록시가 될 수 있습니다. HTTPS 프록시의 경우 인증서 유효성 검사를 건너뛰어 프록시 서버에서 자체 서명된 인증서를 사용할 수 있습니다. 인증이 활성화된 프록시 서버는 지원되지 않습니다.	
nfsMountOptions	NFS 마운트 옵션에 대한 세부적인 제어.	"nfsvers=3"
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
serviceLevel	새로운 볼륨에 대한 CVS-Performance 또는 CVS 서비스 수준입니다. CVS-Performance 값은 다음과 같습니다. standard , premium , 또는 extreme . CVS 값은 standardsw 또는 zoneredundantstandardsw .	CVS-Performance의 기본값은 "표준"입니다. CVS 기본값은 "standardsw"입니다.
network	Cloud Volumes Service 볼륨에 사용되는 Google Cloud 네트워크입니다.	"기본"
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예, \{"api":false, "method":true} . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
allowedTopologies	지역 간 액세스를 활성화하려면 StorageClass 정의를 다음과 같이 합니다. allowedTopologies 모든 지역을 포함해야 합니다. 예를 들어: - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

볼륨 프로비저닝 옵션

기본 볼륨 프로비저닝을 제어할 수 있습니다. defaults 구성 파일의 섹션.

매개변수	설명	기본
exportRule	새로운 볼륨에 대한 내보내기 규칙. CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷의 조합을 쉼표로 구분한 목록이어야 합니다.	"0.0.0.0/0"

매개변수	설명	기본
`snapshotDir`	에 대한 액세스 `.snapshot` 예배 규칙서	"거짓"
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 백분율	"" (CVS 기본값 0 허용)
`size`	새로운 볼륨의 크기. CVS-Performance 최소값은 100GiB입니다. CVS 최소 크기는 1GiB입니다.	CVS-Performance 서비스 유형은 기본적으로 "100GiB"로 설정됩니다. CVS 서비스 유형은 기본값을 설정하지 않지만 최소 1GiB가 필요합니다.

CVS-Performance 서비스 유형 예

다음 예제에서는 CVS-Performance 서비스 유형에 대한 샘플 구성을 제공합니다.

예 **1:** 최소 구성

이는 기본 "표준" 서비스 수준을 갖춘 기본 CVS-Performance 서비스 유형을 사용하는 최소 백엔드 구성입니다.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: "012345678901"
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: <id_value>
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: "123456789012345678901"
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

예 **2:** 서비스 수준 구성

이 샘플은 서비스 수준, 볼륨 기본값을 포함한 백엔드 구성 옵션을 보여줍니다.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

예제 **3:** 가상 풀 구성

이 샘플은 다음을 사용합니다. `storage` 가상 풀을 구성하려면 `StorageClasses` 그것들을 다시 언급하는 것. 참조하다스토리지 클래스 정의 저장 클래스가 어떻게 정의되었는지 확인하세요.

여기서 모든 가상 풀에 대한 특정 기본값이 설정됩니다. `snapshotReserve` 5%에서 그리고 `exportRule` 0.0.0.0/0으로. 가상 풀은 다음에 정의됩니다. `storage` 부분. 각 개별 가상 풀은 자체적으로 정의합니다. `serviceLevel` 일부 풀은 기본값을 덮어씁니다. 가상 풀 레이블은 다음을 기준으로 풀을 구별하는 데 사용되었습니다. `performance` 그리고 `protection` .

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
```

```
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
  serviceLevel: standard
```

스토리지 클래스 정의

다음 StorageClass 정의는 가상 풀 구성 예제에 적용됩니다. 사용 중 `parameters.selector` 각 StorageClass에 대해 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

스토리지 클래스 예제

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
```

```
  selector: performance=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: protection=extra
allowVolumeExpansion: true
```

- 첫 번째 StorageClass(`cvs-extreme-extra-protection`)는 첫 번째 가상 풀에 매핑됩니다. 이 풀은 스냅샷 리저브가 10%로 극한의 성능을 제공하는 유일한 풀입니다.
- 마지막 StorageClass(`cvs-extra-protection`)는 10%의 스냅샷 예약을 제공하는 모든 스토리지 풀을 호출합니다. Trident 어떤 가상 풀을 선택할지 결정하고 스냅샷 예약 요구 사항이 충족되는지 확인합니다.

CVS 서비스 유형 예시

다음 예제에서는 CVS 서비스 유형에 대한 샘플 구성을 제공합니다.

예 **1:** 최소 구성

이는 다음을 사용하는 최소 백엔드 구성입니다. `storageClass` CVS 서비스 유형 및 기본값을 지정하려면 `standardsw` 서비스 수준.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

예 **2:** 스토리지 풀 구성

이 샘플 백엔드 구성은 다음을 사용합니다. `storagePools` 스토리지 풀을 구성하려면.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

다음은 무엇인가요**?**

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

NetApp HCI 또는 SolidFire 백엔드 구성

Trident 설치로 Element 백엔드를 만들고 사용하는 방법을 알아보세요.

요소 드라이버 세부 정보

Trident 다음을 제공합니다. `solidfire-san` 클러스터와 통신하기 위한 저장 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

그만큼 `solidfire-san` 저장 드라이버는 파일 및 블록 볼륨 모드를 지원합니다. 를 위해 `Filesystem` volumeMode, Trident 볼륨을 생성하고 파일 시스템을 생성합니다. 파일 시스템 유형은 StorageClass에 의해 지정됩니다.

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`solidfire-san`	iSCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템이 없습니다. 원시 블록 장치.
`solidfire-san`	iSCSI	파일 시스템	RWO, RWOP	`xfs`, `ext3` , `ext4`

시작하기 전에

Element 백엔드를 생성하려면 다음이 필요합니다.

- Element 소프트웨어를 실행하는 지원되는 저장 시스템입니다.
- 볼륨을 관리할 수 있는 NetApp HCI/ SolidFire 클러스터 관리자 또는 테넌트 사용자에 대한 자격 증명입니다.
- 모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 참조하다"워커 노드 준비 정보" .

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	항상 "solidfire-san"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	"solidfire_" + 스토리지(iSCSI) IP 주소
`Endpoint`	테넌트 자격 증명을 사용한 SolidFire 클러스터용 MVIP	

매개변수	설명	기본
SVIP	스토리지(iSCSI) IP 주소 및 포트	
labels	볼륨에 적용할 임의의 JSON 형식 레이블 집합입니다.	""
TenantName	사용할 테넌트 이름(찾을 수 없는 경우 생성)	
InitiatorIFace	iSCSI 트래픽을 특정 호스트 인터페이스로 제한	"기본"
UseCHAP	CHAP를 사용하여 iSCSI를 인증합니다. Trident CHAP를 사용합니다.	true
AccessGroups	사용할 액세스 그룹 ID 목록	"trident"라는 액세스 그룹의 ID를 찾습니다.
Types	QoS 사양	
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}	널

사용하지 마십시오 `debugTraceFlags` 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.

예 **1:** 백엔드 구성 `solidfire-san` **3**가지 볼륨 유형을 갖춘 드라이버

이 예제에서는 CHAP 인증을 사용하고 특정 QoS 보장을 갖춘 세 가지 볼륨 유형을 모델링하는 백엔드 파일을 보여줍니다. 그러면 각각을 사용하기 위해 저장소 클래스를 정의할 가능성이 가장 높습니다. `IOPS` 저장 클래스 매개변수.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

예 **2:** 백엔드 및 스토리지 클래스 구성 `solidfire-san` 가상 풀이 있는 드라이버

이 예에서는 가상 풀과 이를 참조하는 StorageClass로 구성된 백엔드 정의 파일을 보여줍니다.

Trident 프로비저닝 시 스토리지 풀에 있는 레이블을 백엔드 스토리지 LUN에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값이 설정되어 있습니다. `type` 실버에서. 가상 풀은 다음에 정의됩니다. `storage` 부분. 이 예에서 일부 스토리지 풀은 자체 유형을 설정하고, 일부 풀은 위에 설정된 기본값을 재정의합니다.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
```

```
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

다음 StorageClass 정의는 위의 가상 풀을 참조합니다. 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

첫 번째 StorageClass(`solidfire-gold-four`)은 첫 번째 가상 풀에 매핑됩니다. 이것은 골드 성능을 제공하는 유일한 풀입니다. `Volume Type QoS` 금의. 마지막 StorageClass(`solidfire-silver`)은 실버 성능을 제공하는 모든 스토리지 풀을 호출합니다. Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

더 많은 정보를 찾아보세요

- "볼륨 액세스 그룹"

ONTAP SAN 드라이버

ONTAP SAN 드라이버 개요

ONTAP 및 Cloud Volumes ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보세요.

ONTAP SAN 드라이버 세부 정보

Trident ONTAP 클러스터와 통신하기 위해 다음과 같은 SAN 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	FC를 통한 iSCSI SCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san`	FC를 통한 iSCSI SCSI	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`
`ontap-san`	NVMe/TCP 참조하다NVMe/TCP에 대한 추가 고려 사항 .	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san`	NVMe/TCP 참조하다NVMe/TCP에 대한 추가 고려 사항 .	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san-economy`	iSCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san-economy`	iSCSI	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`

- 사용 `ontap-san-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" .
- 사용 `ontap-nas-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 그리고 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마세요 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 모빌리티가 필요할 것으로 예상되는 경우
- NetApp ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 증가를 사용하지 않는 것을 권장합니다. 이 문제를 해결하기 위해 Trident 스냅샷 예약 사용을 지원하고 Flexvol 볼륨을 그에 따라 확장합니다.

사용자 권한

Trident 일반적으로 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. Amazon FSx for NetApp ONTAP 배포의 경우 Trident 클러스터를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. 그만큼 `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

당신이 사용하는 경우 `limitAggregateUsage` 매개변수, 클러스터 관리자 권한이 필요합니다. Trident 와 함께 Amazon FSx for NetApp ONTAP 사용하는 경우 `limitAggregateUsage` 매개변수는 작동하지 않습니다 `vsadmin` 그리고 `fsxadmin` 사용자 계정. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만드는 것이 가능하지만 권장하지는 않습니다. Trident 의 새로운 릴리스 대부분은 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

NVMe/TCP에 대한 추가 고려 사항

Trident 다음을 사용하여 NVMe(Non-Volatile Memory Express) 프로토콜을 지원합니다. `ontap-san` 운전자 포함:

- IPv6
- NVMe 볼륨의 스냅샷 및 복제본
- NVMe 볼륨 크기 조정
- Trident 외부에서 생성된 NVMe 볼륨을 가져와서 Trident 에서 수명 주기를 관리할 수 있도록 합니다.
- NVMe 네이티브 멀티패싱

- K8s 노드의 정상적 또는 비정상적 종료(24.06)

Trident 다음을 지원하지 않습니다.

- NVMe에서 기본적으로 지원되는 DH-HMAC-CHAP
- 장치 매퍼(DM) 다중 경로
- LUKS 암호화

NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다.

ONTAP SAN 드라이버로 백엔드 구성을 준비합니다**.**

ONTAP SAN 드라이버로 ONTAP 백엔드를 구성하기 위한 요구 사항과 인증 옵션을 이해합니다.

요구 사항

모든 ONTAP 백엔드의 경우 Trident 최소한 하나의 집계가 SVM에 할당되어야 합니다.

"ASA r2 시스템"다른 ONTAP 시스템(ASA, AFF 및 FAS)과 저장 계층 구현 방식이 다릅니다. ASA r2 시스템에서는 집계 대신 스토리지 가용성 영역이 사용됩니다. 참조하다"이것" ASA r2 시스템에서 SVM에 집계를 할당하는 방법에 대한 지식 기반 문서입니다.

두 개 이상의 드라이버를 실행하고, 하나 또는 다른 하나를 가리키는 스토리지 클래스를 생성할 수도 있습니다. 예를 들어 다음을 구성할 수 있습니다. `san-dev` 를 사용하는 클래스 `ontap-san` 운전자와 `san-default` 를 사용하는 클래스 `ontap-san-economy` 하나.

모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 참조하다 "워커 노드 준비" 자세한 내용은.

ONTAP 백엔드 인증

Trident ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름과 비밀번호입니다. 다음과 같은 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다. `admin` 또는 `vsadmin` ONTAP 버전과의 최대 호환성을 보장합니다.
- 인증서 기반: Trident 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 여기서 백엔드 정의에는 클라이언트 인증서, 키, 신뢰할 수 있는 CA 인증서(사용되는 경우)의 Base64 인코딩 값이 포함되어야 합니다(권장).

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 사이를 전환할 수 있습니다. 하지만 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

*자격 증명과 인증서*를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. 다음과 같은 표준 사전 정의된 역할을 활용하는 것이 좋습니다. `admin` 또는 `vsadmin` . 이를 통해 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 공개할 수 있는 향후 ONTAP 릴리스와의 호환성이 보장됩니다. Trident 사용하면 사용자 정의 보안 로그인 역할을 만들어 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 곳이라는 점을 명심하세요. 백엔드가 생성된 후 사용자 이름/비밀번호는 Base64로 인코딩되어 Kubernetes 비밀로 저장됩니다. 백엔드를 만들거나 업데이트하는 것은 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행해야 하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

새로운 백엔드와 기존 백엔드는 인증서를 사용하고 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 개의 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64로 인코딩된 값입니다.
- clientPrivateKey: 연관된 개인 키의 Base64 인코딩된 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않으면 무시할 수 있습니다.

일반적인 작업 흐름은 다음 단계로 구성됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성할 때, 인증할 ONTAP 사용자로 일반 이름(CN)을 설정합니다.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 이 문제는 이미 스토리지 관리자가 처리했을 수도 있습니다. 신뢰할 수 있는 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서와 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 지원되는지 확인하세요. `cert` 인증방법.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. < ONTAP 관리 LIF> 및 <vserver 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법 업데이트 또는 자격 증명 순환

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 순환할 수 있습니다. 이는 양방향으로 작동합니다. 사용자 이름/비밀번호를 사용하는 백엔드는 인증서를 사용하도록 업데이트할 수 있고, 인증서를 활용하는 백엔드는 사용자 이름/비밀번호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새로운 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수를 포함하는 업데이트된 backend.json 파일을 사용하여 실행합니다. `tridentctl backend update` .

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

비밀번호를 순환할 때 스토리지 관리자는 먼저 ONTAP 에서 사용자의 비밀번호를 업데이트해야 합니다. 이어서 백엔드 업데이트가 진행됩니다. 인증서를 순환할 때 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 그런 다음 백엔드를 업데이트하여 새 인증서를 사용하고, 그 후 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 만들어진 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공하면 Trident ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident 에 대한 사용자 정의 ONTAP 역할 생성

Trident 에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용하지 않아도 되도록 최소한의 권한으로 ONTAP 클러스터 역할을 만들 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident 사용자가 만든 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

참조하다"Trident 사용자 정의 역할 생성기" Trident 사용자 정의 역할 생성에 대한 자세한 내용은 다음을 참조하세요.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 만듭니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자에 대한 사용자 이름을 만듭니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할을 매핑합니다.

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

시스템 관리자 사용

ONTAP 시스템 관리자에서 다음 단계를 수행합니다.

1. 사용자 정의 역할 만들기:
   a. 클러스터 수준에서 사용자 지정 역할을 만들려면 *클러스터 > 설정*을 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 만들려면 저장소 **>** 저장소 **VM > `required SVM` >** 설정 **>** 사용자 및 역할.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. *역할*에서 *+추가*를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. * Trident 사용자에게 역할 매핑*: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에 있는 추가 아이콘 *+*을 선택합니다.
   b. 필요한 사용자 이름을 선택하고, 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭하세요.

자세한 내용은 다음 페이지를 참조하세요.

- "ONTAP 관리를 위한 사용자 정의 역할"또는"사용자 정의 역할 정의"
- "역할 및 사용자 작업"

양방향 **CHAP**를 사용하여 연결 인증

Trident 양방향 CHAP를 사용하여 iSCSI 세션을 인증할 수 있습니다. `ontap-san` 그리고 `ontap-san-economy` 운전자. 이를 위해서는 다음을 활성화해야 합니다. `useCHAP` 백엔드 정의에 옵션을 추가하세요. 설정 시 `true` Trident SVM의 기본 이니시에이터 보안을 양방향 CHAP로 구성하고 백엔드 파일에서 사용자 이름과 비밀번호를 설정합니다.

NetApp 양방향 CHAP를 사용하여 연결을 인증할 것을 권장합니다. 다음 샘플 구성을 참조하세요.

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

그만큼 `useCHAP` 매개변수는 한 번만 구성할 수 있는 부울 옵션입니다. 기본적으로 false로 설정됩니다. true로 설정한 후에는 false로 설정할 수 없습니다.

또한 `useCHAP=true` , 그 `chapInitiatorSecret` , `chapTargetInitiatorSecret` , `chapTargetUsername` , 그리고 `chapUsername` 필드는 백엔드 정의에 포함되어야 합니다. 백엔드가 생성된 후 다음을 실행하여 비밀을 변경할 수 있습니다. `tridentctl update` .

작동 원리

설정하여 `useCHAP` true로 설정하면 스토리지 관리자가 Trident 스토리지 백엔드에서 CHAP를 구성하도록 지시합니다. 여기에는 다음이 포함됩니다.

- SVM에 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 없음(기본적으로 설정됨)이고 볼륨에 이미 존재하는 LUN이 없는 경우 Trident 기본 보안 유형을 다음과 같이 설정합니다. `CHAP` CHAP 이니시에이터와 대상 사용자 이름 및 비밀번호를 구성합니다.
  - SVM에 LUN이 포함되어 있으면 Trident SVM에서 CHAP를 활성화하지 않습니다. 이렇게 하면 SVM에 이미 있는 LUN에 대한 액세스가 제한되지 않습니다.
- CHAP 이니시에이터와 대상 사용자 이름 및 비밀번호를 구성합니다. 이러한 옵션은 백엔드 구성에서 지정해야 합니다(위에 표시된 대로).

백엔드가 생성된 후 Trident 해당 백엔드를 생성합니다. `tridentbackend` CRD를 사용하여 CHAP 비밀과 사용자 이름을 Kubernetes 비밀로 저장합니다. 이 백엔드에서 Trident 가 생성한 모든 PV는 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 순환하고 백엔드를 업데이트합니다.

CHAP 매개변수를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있습니다. `backend.json` 파일. 이렇게 하려면 CHAP 비밀을 업데이트하고 다음을 사용해야 합니다. `tridentctl update` 이러한 변경 사항을 반영하기 위한 명령입니다.

백엔드의 CHAP 비밀을 업데이트할 때는 다음을 사용해야 합니다. `tridentctl` 백엔드를 업데이트합니다. Trident 이러한 변경 사항을 선택할 수 없으므로 ONTAP CLI 또는 ONTAP 시스템 관리자를 사용하여 스토리지 클러스터의 자격 증명을 업데이트하지 마십시오.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

기존 연결은 영향을 받지 않으며, SVM에서 Trident 가 자격 증명을 업데이트하면 계속 활성 상태를 유지합니다. 새로운 연결은 업데이트된 자격 증명을 사용하고 기존 연결은 계속 활성 상태를 유지합니다. 이전 PV의 연결을 끊었다가 다시 연결하면 업데이트된 자격 증명을 사용하게 됩니다.

ONTAP SAN 구성 옵션 및 예

Trident 설치로 ONTAP SAN 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClass에 매핑하기 위한 세부 정보를 제공합니다.

"ASA r2 시스템"다른 ONTAP 시스템(ASA, AFF, FAS)과 저장 계층 구현 방식이 다릅니다. 이러한 변화는 표기된 특정 매개변수의 사용에 영향을 미칩니다. "ASA r2 시스템과 다른 ONTAP 시스템 간의 차이점에 대해 자세히 알아보세요.".

오직 `ontap-san` 드라이버(iSCSI 및 NVMe/TCP 프로토콜 포함)는 ASA r2 시스템에서 지원됩니다.

Trident 백엔드 구성에서는 시스템이 ASA r2라고 지정할 필요가 없습니다. 선택할 때 `ontap-san` 로서 `storageDriverName` Trident ASA r2 또는 기존 ONTAP 시스템을 자동으로 감지합니다. 아래 표에 나와 있는 것처럼 일부 백엔드 구성 매개변수는 ASA r2 시스템에 적용할 수 없습니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-san`또는 `ontap-san-economy`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 원활한 MetroCluster 전환을 위해서는 다음을 참조하세요 .MetroCluster 예제 . "vsadmin" 자격 증명을 사용하는 경우 `managementLIF` SVM의 자격 증명이어야 합니다. "관리자" 자격 증명을 사용하는 경우 `managementLIF` 클러스터의 것이어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	프로토콜 LIF의 IP 주소. Trident IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . **iSCSI**에 대해서는 지정하지 마세요**.** Trident 사용"ONTAP 선택적 LUN 맵" 다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. 경고가 생성됩니다. `dataLIF` 명확하게 정의되어 있습니다. 메트로클러스터는 생략합니다**.** 를 참조하십시오MetroCluster 예제 .	SVM에 의해 파생됨
`svm`	사용할 스토리지 가상 머신 **Metrocluster**의 경우 생략 를 참조하십시오MetroCluster 예제 .	SVM의 경우 파생됨 `managementLIF` 지정됨

매개변수	설명	기본
`useCHAP`	CHAP를 사용하여 ONTAP SAN 드라이버에 대한 iSCSI를 인증합니다[부울]. 로 설정 `true` Trident 백엔드에 제공된 SVM에 대한 기본 인증으로 양방향 CHAP를 구성하고 사용하도록 합니다. 참조하다 "ONTAP SAN 드라이버로 백엔드 구성을 준비합니다." 자세한 내용은. **FCP** 또는 **NVMe/TCP**에서는 지원되지 않습니다.	`false`
`chapInitiatorSecret`	CHAP 개시자 비밀. 필요한 경우 `useCHAP=true`	""
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`chapTargetInitiatorSecret`	CHAP 대상 개시자 비밀. 필요한 경우 `useCHAP=true`	""
`chapUsername`	수신 사용자 이름. 필요한 경우 `useCHAP=true`	""
`chapTargetUsername`	대상 사용자 이름. 필요한 경우 `useCHAP=true`	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됩니다.	""
`username`	ONTAP 클러스터와 통신하려면 사용자 이름이 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
`password`	ONTAP 클러스터와 통신하려면 비밀번호가 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
`svm`	사용할 스토리지 가상 머신	SVM의 경우 파생됨 `managementLIF` 지정됨
`storagePrefix`	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 나중에 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새로운 백엔드를 만들어야 합니다.	`trident`

매개변수	설명	기본
`aggregate`	프로비저닝을 위한 집계(선택 사항, 설정된 경우 SVM에 할당해야 함). 를 위해 `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당되지 않은 경우 사용 가능한 모든 집계를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 집계가 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계를 구성한 경우, 집계의 이름이 변경되거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 SVM에 있는 집계로 변경하거나 집계를 완전히 제거해야 합니다. • ASA r2 시스템에 대해서는 지정하지 마세요*.	""
`limitAggregateUsage`	사용량이 이 백분율을 초과하면 프로비저닝에 실패합니다. Amazon FSx for NetApp ONTAP 백엔드를 사용하는 경우 지정하지 마십시오. `limitAggregateUsage` . 제공된 `fsxadmin` 그리고 `vsadmin` Trident 사용하여 집계 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다. * ASA r2 시스템에 대해서는 지정하지 마세요*.	"" (기본적으로 적용되지 않음)
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	Flexvol당 최대 LUN은 [50, 200] 범위 내에 있어야 합니다.	`100`
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}는 문제를 해결하고 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마세요.	`null`

매개변수	설명	기본
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. `useREST`설정 시 `true` , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. `false` Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. `ontapi` 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. `vsadmin` 그리고 `cluster-admin` 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST` 로 설정됩니다 `true` 기본적으로; 변경 `useREST` 에게 `false` ONTAPI(ZAPI) 호출을 사용합니다. `useREST`NVMe/TCP에 완벽하게 적합합니다. NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다. *지정된 경우 항상 다음으로 설정됩니다. true ASA r2 시스템*용.	true`ONTAP 9.15.1 이상인 경우, 그렇지 않은 경우 `false.
sanType	선택에 사용 iscsi iSCSI의 경우, nvme NVMe/TCP 또는 fcp FC(Fibre Channel)를 통한 SCSI의 경우.	`iscsi`비어있는 경우
formatOptions	사용 formatOptions 명령줄 인수를 지정하려면 mkfs 볼륨이 포맷될 때마다 적용되는 명령입니다. 이를 통해 사용자의 선호도에 따라 볼륨을 포맷할 수 있습니다. 장치 경로를 제외하고 mkfs 명령 옵션과 비슷하게 formatOptions를 지정해야 합니다. 예: "-E nodiscard" 지원됨 **ontap-san** 그리고 **ontap-san-economy** **iSCSI** 프로토콜을 사용하는 드라이버. 또한 **iSCSI** 및 **NVMe/TCP** 프로토콜을 사용하는 **ASA r2** 시스템에서도 지원됩니다.	
limitVolumePoolSize	ontap-san-economy 백엔드에서 LUN을 사용할 때 요청 가능한 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본
`denyNewVolumePools`	제한하다 `ontap-san-economy` 백엔드가 LUN을 포함하기 위해 새로운 FlexVol 볼륨을 생성하지 못하도록 합니다. 새로운 PV를 프로비저닝하는 데는 기존 Flexvol만 사용됩니다.	

formatOptions 사용에 대한 권장 사항

Trident 포맷 과정을 신속하게 진행하기 위해 다음 옵션을 권장합니다.

-E 삭제 안 함**:**

- 유지, mkfs 시점에 블록을 삭제하려고 시도하지 마세요(처음에 블록을 삭제하는 것은 솔리드 스테이트 장치와 스파스/씬 프로비저닝 스토리지에서 유용합니다). 이는 더 이상 사용되지 않는 옵션인 "-K"를 대체하며 모든 파일 시스템(xfs, ext3, ext4)에 적용할 수 있습니다.

Active Directory 자격 증명을 사용하여 백엔드 **SVM**에 **Trident** 인증

Active Directory(AD) 자격 증명을 사용하여 백엔드 SVM에 인증하도록 Trident 구성할 수 있습니다. AD 계정이 SVM에 액세스하려면 먼저 클러스터 또는 SVM에 대한 AD 도메인 컨트롤러 액세스를 구성해야 합니다. AD 계정으로 클러스터를 관리하려면 도메인 터널을 만들어야 합니다. 참조하다 "ONTAP 에서 Active Directory 도메인 컨트롤러 액세스 구성" 자세한 내용은.

단계

1. 백엔드 SVM에 대한 DNS(도메인 이름 시스템) 설정을 구성합니다.

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Active Directory에서 SVM에 대한 컴퓨터 계정을 만들려면 다음 명령을 실행하세요.

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. 이 명령을 사용하여 클러스터 또는 SVM을 관리할 AD 사용자 또는 그룹을 만듭니다.

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Trident 백엔드 구성 파일에서 다음을 설정합니다. `username` 그리고 `password` 각각 AD 사용자 또는 그룹 이름과 비밀번호에 대한 매개변수입니다.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. `defaults` 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
`spaceAllocation`	LUN에 대한 공간 할당	"true" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
`spaceReserve`	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움). 설정 **none ASA r2** 시스템용.	"없음"
`snapshotPolicy`	사용할 스냅샷 정책입니다. *설정 `none` ASA r2 시스템*용.	"없음"
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용해야 하며, 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하세요.	""
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 비율입니다. * ASA r2 시스템에 대해서는 지정하지 마세요*.	"0"이면 `snapshotPolicy` "없음"이고, 그렇지 않으면 ""
`splitOnClone`	생성 시 부모로부터 복제본을 분할합니다.	"거짓"
`encryption`	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. `false` . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	"false" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
`luksEncryption`	LUKS 암호화를 활성화합니다. 참조하다"Linux Unified Key Setup(LUKS) 사용" .	"" *로 설정 `false` ASA r2 시스템*용.
`tieringPolicy`	"없음"을 사용하는 계층화 정책 * ASA r2 시스템에는 지정하지 마세요*.	
`nameTemplate`	사용자 정의 볼륨 이름을 만드는 템플릿입니다.	""

볼륨 프로비저닝 예시

기본값이 정의된 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

다음을 사용하여 생성된 모든 볼륨의 경우 `ontap-san` 드라이버, Trident LUN 메타데이터를 수용하기 위해 FlexVol 에 10%의 추가 용량을 추가합니다. LUN은 PVC에서 사용자가 요청한 정확한 크기로 프로비저닝됩니다. Trident FlexVol 에 10%를 추가합니다(ONTAP 에서는 사용 가능한 크기로 표시됨). 이제 사용자는 요청한 사용 가능한 용량을 얻게 됩니다. 이 변경 사항은 사용 가능한 공간이 완전히 활용되지 않는 한 LUN이 읽기 전용이 되는 것을 방지합니다. 이는 ontap-san-economy에는 적용되지 않습니다.

정의하는 백엔드의 경우 `snapshotReserve` Trident 다음과 같이 볼륨의 크기를 계산합니다.

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Trident FlexVol 에 추가하는 10%입니다. 을 위한 `snapshotReserve` = 5%, PVC 요청 = 5GiB인 경우 총 볼륨 크기는 5.79GiB이고 사용 가능한 크기는 5.5GiB입니다. 그만큼 `volume show` 명령을 실행하면 다음 예와 비슷한 결과가 표시됩니다.

현재 기존 볼륨에 대한 새로운 계산을 사용하려면 크기 조정만이 유일한 방법입니다.

최소 구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

> Trident 와 함께 NetApp ONTAP 에서 Amazon FSx 사용하는 경우 NetApp IP 주소 대신 LIF에 DNS 이름을 지정할 것을 권장합니다.

ONTAP SAN 예시

이는 다음을 사용하는 기본 구성입니다. `ontap-san` 운전사.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

MetroCluster 예제

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다."SVM 복제 및 복구" .

원활한 전환 및 전환을 위해 다음을 사용하여 SVM을 지정하십시오. `managementLIF` 그리고 생략하다 `svm` 매개변수. 예를 들어:

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

ONTAP SAN 경제 사례

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

인증서 기반 인증 예제

이 기본 구성 예에서는 `clientCertificate` , `clientPrivateKey` , 그리고 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)이 채워집니다. `backend.json` 그리고 클라이언트 인증서, 개인 키, 신뢰할 수 있는 CA 인증서의 base64 인코딩 값을 각각 가져옵니다.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

양방향 **CHAP** 예시

이러한 예제는 백엔드를 생성합니다. `useCHAP` 로 설정 `true` .

ONTAP SAN CHAP 예시

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN 경제 **CHAP** 예시

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCP 예제

ONTAP 백엔드에 NVMe로 구성된 SVM이 있어야 합니다. 이는 NVMe/TCP의 기본 백엔드 구성입니다.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

FC(FCP)를 통한 **SCSI** 예제

ONTAP 백엔드에 FC가 구성된 SVM이 있어야 합니다. 이는 FC의 기본 백엔드 구성입니다.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

ontap-san-economy 드라이버에 대한 **formatOptions** 예제

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

가상 풀을 사용한 백엔드의 예

이러한 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 다음과 같은 특정 기본값이 설정됩니다. `spaceReserve` 없음, `spaceAllocation` 거짓이고, `encryption` 거짓. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident "주석" 필드에 프로비저닝 라벨을 설정합니다. FlexVol volume 에 대한 주석이 설정되면 Trident 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이러한 예에서 일부 스토리지 풀은 자체적으로 설정합니다. `spaceReserve` , `spaceAllocation` , 그리고 `encryption` 값이 지정되고 일부 풀은 기본값을 재정의합니다.

ONTAP SAN 예시

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

ONTAP SAN 경제 사례

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
```

```
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

NVMe/TCP 예제

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

백엔드를 **StorageClass**에 매핑

다음 StorageClass 정의는 다음을 참조합니다.가상 풀을 사용한 백엔드의 예 . 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

- 그만큼 `protection-gold` StorageClass는 첫 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이곳은 골드 레벨의 보호를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 그만큼 `protection-not-gold` StorageClass는 두 번째 및 세 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이것들은 금 이외의 보호 수준을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 그만큼 `app-mysqldb` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-san-economy` 백엔드. 이는 mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 그만큼 `protection-silver-creditpoints-20k` StorageClass는 두 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이 풀은 실버 레벨 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 그만큼 `creditpoints-5k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드와 네 번째 가상 풀 `ontap-san-economy` 백엔드. 5000 크레딧 포인트를 제공하는 유일한 풀 서비스입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- 그만큼 `my-test-app-sc` StorageClass는 다음에 매핑됩니다. `testAPP` 가상 풀 `ontap-san` 운전자와 함께 `sanType: nvme` . 이것은 유일한 풀 제공입니다 `testApp` .

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

ONTAP NAS 드라이버

ONTAP NAS 드라이버 개요

ONTAP 및 Cloud Volumes ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보세요.

ONTAP NAS 드라이버 세부 정보

Trident ONTAP 클러스터와 통신하기 위해 다음과 같은 NAS 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-nas`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`
`ontap-nas-economy`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`
`ontap-nas-flexgroup`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`

- 사용 `ontap-san-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" .
- 사용 `ontap-nas-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 그리고 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마세요 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 모빌리티가 필요할 것으로 예상되는 경우
- NetApp ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 증가를 사용하지 않는 것을 권장합니다. 이 문제를 해결하기 위해 Trident 스냅샷 예약 사용을 지원하고 Flexvol 볼륨을 그에 따라 확장합니다.

사용자 권한

Trident 일반적으로 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다.

Amazon FSx for NetApp ONTAP 배포의 경우 Trident 클러스터를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. 그만큼 `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

당신이 사용하는 경우 `limitAggregateUsage` 매개변수, 클러스터 관리자 권한이 필요합니다. Trident 와 함께 Amazon FSx for NetApp ONTAP 사용하는 경우 `limitAggregateUsage` 매개변수는 작동하지 않습니다 `vsadmin` 그리고 `fsxadmin` 사용자 계정. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만드는 것이 가능하지만 권장하지는 않습니다. Trident 의 새로운 릴리스 대부분은 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

ONTAP NAS 드라이버로 백엔드 구성을 준비합니다.

ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항, 인증 옵션 및 내보내기 정책을 이해합니다.

요구 사항

- 모든 ONTAP 백엔드의 경우 Trident 최소한 하나의 집계가 SVM에 할당되어야 합니다.
- 두 개 이상의 드라이버를 실행하고, 하나를 가리키는 스토리지 클래스를 만들 수 있습니다. 예를 들어, 다음을 사용하는 Gold 클래스를 구성할 수 있습니다. `ontap-nas` 드라이버 및 Bronze 클래스를 사용하는 `ontap-nas-economy` 하나.
- 모든 Kubernetes 워커 노드에는 적절한 NFS 도구가 설치되어 있어야 합니다. 참조하다"여기" 자세한 내용은.
- Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 참조하다 SMB 볼륨 프로비저닝 준비 자세한 내용은.

ONTAP 백엔드 인증

Trident ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 이 모드에서는 ONTAP 백엔드에 대한 충분한 권한이 필요합니다. 다음과 같은 사전 정의된 보안 로그인 역할과 연결된 계정을 사용하는 것이 좋습니다. `admin` 또는 `vsadmin` ONTAP 버전과의 최대 호환성을 보장합니다.
- 인증서 기반: 이 모드에서는 Trident ONTAP 클러스터와 통신하려면 백엔드에 인증서가 설치되어 있어야 합니다. 여기서 백엔드 정의에는 클라이언트 인증서, 키, 신뢰할 수 있는 CA 인증서(사용되는 경우)의 Base64 인코딩 값이 포함되어야 합니다(권장).

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 사이를 전환할 수 있습니다. 하지만 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

*자격 증명과 인증서*를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. 다음과 같은 표준 사전 정의된 역할을 활용하는 것이 좋습니다. `admin` 또는 `vsadmin` . 이를 통해 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 공개할 수 있는 향후 ONTAP 릴리스와의 호환성이 보장됩니다. Trident 사용하면 사용자 정의 보안 로그인 역할을 만들어 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 곳이라는 점을 명심하세요. 백엔드가 생성된 후 사용자 이름/비밀번호는 Base64로 인코딩되어 Kubernetes 비밀로 저장됩니다. 백엔드를 생성/업데이트하는 단계는 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행해야 하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

새로운 백엔드와 기존 백엔드는 인증서를 사용하고 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 개의 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64로 인코딩된 값입니다.
- clientPrivateKey: 연관된 개인 키의 Base64 인코딩된 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않으면 무시할 수 있습니다.

일반적인 작업 흐름은 다음 단계로 구성됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성할 때, 인증할 ONTAP 사용자로 일반 이름(CN)을 설정합니다.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 이 문제는 이미 스토리지 관리자가 처리했을 수도 있습니다. 신뢰할 수 있는 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서와 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 지원되는지 확인하세요. `cert` 인증방법.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. < ONTAP 관리 LIF> 및 <vserver 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다. LIF의 서비스 정책이 다음과 같이 설정되어 있는지 확인해야 합니다. `default-data-management` .

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법 업데이트 또는 자격 증명 순환

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 순환할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/비밀번호를 사용하는 백엔드는 인증서를 사용하도록 업데이트할 수 있고, 인증서를 활용하는 백엔드는 사용자 이름/비밀번호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새로운 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수를 포함하는 업데이트된 backend.json 파일을 사용하여 실행합니다. `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

비밀번호를 순환할 때 스토리지 관리자는 먼저 ONTAP 에서 사용자의 비밀번호를 업데이트해야 합니다. 이어서 백엔드 업데이트가 진행됩니다. 인증서를 순환할 때 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 그런 다음 백엔드를 업데이트하여 새 인증서를 사용하고, 그 후 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 만들어진 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공하면 Trident ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident 에 대한 사용자 정의 ONTAP 역할 생성

Trident 에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용하지 않아도 되도록 최소한의 권한으로 ONTAP 클러스터 역할을 만들 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident 사용자가 만든 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

참조하다"Trident 사용자 정의 역할 생성기" Trident 사용자 정의 역할 생성에 대한 자세한 내용은 다음을 참조하세요.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 만듭니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자에 대한 사용자 이름을 만듭니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할을 매핑합니다.

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

시스템 관리자 사용

ONTAP 시스템 관리자에서 다음 단계를 수행합니다.

1. 사용자 정의 역할 만들기:
   a. 클러스터 수준에서 사용자 지정 역할을 만들려면 *클러스터 > 설정*을 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 만들려면 저장소 **>** 저장소 **VM > `required SVM` >** 설정 **>** 사용자 및 역할.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. *역할*에서 *+추가*를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. * Trident 사용자에게 역할 매핑*: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에 있는 추가 아이콘 *+*을 선택합니다.
   b. 필요한 사용자 이름을 선택하고, 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭하세요.

자세한 내용은 다음 페이지를 참조하세요.

- "ONTAP 관리를 위한 사용자 정의 역할"또는"사용자 정의 역할 정의"
- "역할 및 사용자 작업"

NFS 내보내기 정책 관리

Trident NFS 내보내기 정책을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어합니다.

Trident 수출 정책을 사용할 때 두 가지 옵션을 제공합니다.

- Trident 내보내기 정책을 직접 동적으로 관리할 수 있습니다. 이 작동 모드에서 스토리지 관리자는 허용 가능한 IP 주소를 나타내는 CIDR 블록 목록을 지정합니다. Trident 게시 시점에 해당 범위에 해당하는 노드 IP를 자동으로 내보내기 정책에 추가합니다. 또는 CIDR이 지정되지 않은 경우 볼륨이 게시되는 노드에서 발견된 모든 글로벌 범위 유니캐스트 IP가 내보내기 정책에 추가됩니다.
- 스토리지 관리자는 내보내기 정책을 만들고 규칙을 수동으로 추가할 수 있습니다. 구성에서 다른 내보내기 정책 이름이 지정되지 않는 한 Trident 기본 내보내기 정책을 사용합니다.

동적으로 수출 정책을 관리합니다

Trident ONTAP 백엔드에 대한 내보내기 정책을 동적으로 관리하는 기능을 제공합니다. 이를 통해 스토리지 관리자는 명시적 규칙을 수동으로 정의하는 대신, 작업자 노드 IP에 허용되는 주소 공간을 지정할 수 있습니다. 이를 통해 내보내기 정책 관리가 대폭 간소화됩니다. 내보내기 정책을 수정하더라도 더 이상 스토리지 클러스터에 대한 수동 개입이 필요하지 않습니다. 또한 이를 통해 볼륨을 마운트하고 지정된 범위 내의 IP를 가진 작업자 노드에만 스토리지 클러스터에 대한 액세스를 제한하여 세분화되고 자동화된 관리를 지원합니다.

> 동적 내보내기 정책을 사용할 때는 NAT(네트워크 주소 변환)를 사용하지 마세요. NAT를 사용하면 스토리지 컨트롤러는 실제 IP 호스트 주소가 아닌 프런트엔드 NAT 주소를 확인하므로 내보내기 규칙에서 일치하는 항목이 없으면 액세스가 거부됩니다.

예

사용해야 하는 구성 옵션은 두 가지가 있습니다. 백엔드 정의의 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

> 이 기능을 사용하려면 SVM의 루트 접합에 노드 CIDR 블록을 허용하는 내보내기 규칙(예: 기본 내보내기 정책)이 포함된 이전에 생성된 내보내기 정책이 있는지 확인해야 합니다. Trident 에 SVM을 전용으로 지정하려면 항상 NetApp 권장하는 모범 사례를 따르세요.

위의 예를 사용하여 이 기능이 작동하는 방식을 설명하겠습니다.

- `autoExportPolicy`로 설정됩니다 `true`. 이는 Trident 이 백엔드로 프로비저닝된 각 볼륨에 대한 내보내기 정책을 생성함을 나타냅니다. `svm1` SVM을 사용하여 규칙 추가 및 삭제를 처리합니다. `autoexportCIDRs` 주소 블록. 볼륨이 노드에 연결될 때까지 해당 볼륨은 원치 않는 볼륨 액세스를 방지하는 규칙이 없는 빈 내보내기 정책을 사용합니다. 볼륨이 노드에 게시되면 Trident 지정된 CIDR 블록 내의 노드 IP를 포함하는 기본 qtree와 동일한 이름으로 내보내기 정책을 생성합니다. 이러한 IP는 부모 FlexVol volume 에서 사용하는 내보내기 정책에도 추가됩니다.

◦ 예를 들어:

- 백엔드 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
- `autoExportPolicy`로 설정 `true`
- 저장 접두사 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c라는 qtree는 FlexVol 에 대한 내보내기 정책을 생성합니다. `trident-403b5326-8482-40db96d0-d83fb3f4daec` , qtree에 대한 내보내기 정책 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` , 그리고 이름이 비어 있는 내보내기 정책 `trident_empty` SVM에 대해. FlexVol 내보내기 정책에 대한 규칙은 qtree 내보내기 정책에 포함된 모든 규칙의 상위 집합입니다. 빈 내보내기 정책은 첨부되지 않은 모든 볼륨에서 재사용됩니다.

- `autoExportCIDRs`주소 블록 목록이 포함되어 있습니다. 이 필드는 선택 사항이며 기본값은 ["0.0.0.0/0", "::/0"]입니다. 정의되지 않은 경우, Trident 작업자 노드에서 발견된 모든 글로벌 범위 유니캐스트 주소를 게시물과 함께 추가합니다.

이 예에서는 `192.168.0.0/24` 주소 공간이 제공됩니다. 이는 게시된 내용이 있는 이 주소 범위에 속하는 Kubernetes 노드 IP가 Trident 에서 생성하는 내보내기 정책에 추가됨을 나타냅니다. Trident 실행되는 노드를 등록하면 노드의 IP 주소를 검색하여 제공된 주소 블록과 비교합니다. `autoExportCIDRs` 게시 시점에 IP를 필터링한 후 Trident 게시 대상 노드의 클라이언트 IP에 대한 내보내기 정책 규칙을 만듭니다.

업데이트할 수 있습니다 `autoExportPolicy` 그리고 `autoExportCIDRs` 백엔드를 만든 후 백엔드에 대해 알아보세요. 자동으로 관리되는 백엔드에 새로운 CIDR을 추가하거나 기존 CIDR을 삭제할 수 있습니다. CIDR을 삭제할 때는 기존 연결이 끊어지지 않도록 주의하세요. 비활성화하도록 선택할 수도 있습니다. `autoExportPolicy` 백엔드의 경우 수동으로 만든 내보내기 정책으로 돌아갑니다. 이렇게 하려면 다음을 설정해야 합니다. `exportPolicy` 백엔드 구성의 매개변수입니다.

Trident 백엔드를 생성하거나 업데이트한 후에는 다음을 사용하여 백엔드를 확인할 수 있습니다. `tridentctl` 또는 해당 `tridentbackend` CRD:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

노드가 제거되면 Trident 모든 내보내기 정책을 확인하여 노드에 해당하는 액세스 규칙을 제거합니다. Trident 관리형 백엔드의 내보내기 정책에서 이 노드 IP를 제거하여 클러스터의 새 노드에서 이 IP를 재사용하지 않는 한 악성 마운트를 방지합니다.

기존 백엔드의 경우 백엔드를 업데이트합니다. `tridentctl update backend` Trident 자동으로 수출 정책을 관리하도록 보장합니다. 이렇게 하면 필요할 때 백엔드의 UUID와 qtree 이름을 따서 명명된 두 개의 새로운 내보내기 정책이 생성됩니다. 백엔드에 있는 볼륨은 마운트 해제 후 다시 마운트되면 새로 생성된 내보내기 정책을 사용합니다.

> ⓘ 자동 관리형 내보내기 정책이 있는 백엔드를 삭제하면 동적으로 생성된 내보내기 정책도 삭제됩니다. 백엔드를 다시 만들면 새로운 백엔드로 처리되어 새로운 내보내기 정책이 생성됩니다.

라이브 노드의 IP 주소가 업데이트되면 노드에서 Trident Pod를 다시 시작해야 합니다. 그러면 Trident 해당 IP 변경 사항을 반영하기 위해 관리하는 백엔드에 대한 내보내기 정책을 업데이트합니다.

SMB 볼륨 프로비저닝 준비

약간의 추가 준비로 다음을 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. `ontap-nas` 운전자.

> ⓘ SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성하여 다음을 생성해야 합니다. `ontap-nas-economy` ONTAP 온프레미스 클러스터를 위한 SMB 볼륨. 이러한 프로토콜 중 하나라도 구성하지 못하면 SMB 볼륨 생성이 실패합니다.

> ⓘ `autoExportPolicy`SMB 볼륨에서는 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하려면 다음이 필요합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명을 포함하는 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

단계

1. 온프레미스 ONTAP 의 경우 선택적으로 SMB 공유를 만들 수 있으며, Trident 만들어줄 수도 있습니다.

   Amazon FSx for ONTAP 에는 SMB 공유가 필요합니다.

   다음 두 가지 방법 중 하나를 사용하여 SMB 관리자 공유를 생성할 수 있습니다."Microsoft 관리 콘솔" 공유 폴더 스냅인 또는 ONTAP CLI 사용. ONTAP CLI를 사용하여 SMB 공유를 생성하려면:

   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 만듭니다.

      그만큼 `vserver cifs share create` 이 명령은 공유 생성 중에 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연관된 SMB 공유를 만듭니다.

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. 공유가 생성되었는지 확인하세요.

```
vserver cifs share show -share-name share_name
```

   참조하다"SMB 공유 만들기" 자세한 내용은 다음을 참조하세요.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대해서는 다음을 참조하세요."FSx for ONTAP 구성 옵션 및 예제" .

매개변수	설명	예
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름, Trident SMB 공유를 만들 수 있도록 하는 이름, 또는 매개변수를 비워 두면 볼륨에 대한 일반 공유 액세스를 방지할 수 있습니다. 이 매개변수는 온프레미스 ONTAP 의 경우 선택 사항입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필수이므로 비워둘 수 없습니다.	smb-share
nasType	설정해야 합니다 **smb .** null인 경우 기본값은 다음과 같습니다. nfs .	smb
securityStyle	새로운 볼륨에 대한 보안 스타일입니다. 설정해야 합니다 **ntfs** 또는 **mixed SMB** 볼륨의 경우**.**	ntfs`또는 `mixed SMB 볼륨의 경우
unixPermissions	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워두어야 합니다**.**	""

보안 **SMB** 활성화

25.06 릴리스부터 NetApp Trident 다음을 사용하여 생성된 SMB 볼륨의 보안 프로비저닝을 지원합니다. ontap-nas 그리고 ontap-nas-economy 백엔드. 보안 SMB가 활성화되면 액세스 제어 목록(ACL)을 사용하여 Active Directory(AD) 사용자 및 사용자 그룹의 공유에 대한 SMB에 대한 제어된 액세스를 제공할 수 있습니다.

기억해야 할 점

- 수입 ontap-nas-economy 볼륨은 지원되지 않습니다.
- 읽기 전용 복제본만 지원됩니다. ontap-nas-economy 볼륨.
- Secure SMB가 활성화된 경우 Trident 백엔드에 언급된 SMB 공유를 무시합니다.
- PVC 주석, 스토리지 클래스 주석 및 백엔드 필드를 업데이트해도 SMB 공유 ACL은 업데이트되지 않습니다.
- 복제 PVC의 주석에 지정된 SMB 공유 ACL은 소스 PVC의 ACL보다 우선합니다.
- 보안 SMB를 활성화하는 동안 유효한 AD 사용자를 제공해야 합니다. 유효하지 않은 사용자는 ACL에 추가되지 않습니다.
- 백엔드, 스토리지 클래스, PVC에서 동일한 AD 사용자에게 서로 다른 권한을 제공하는 경우 권한 우선순위는 PVC, 스토리지 클래스, 백엔드 순입니다.
- 보안 SMB가 지원됩니다. ontap-nas 관리되는 볼륨 가져오기에는 적용되며 관리되지 않는 볼륨 가져오기에는 적용되지 않습니다.

단계

1. 다음 예와 같이 TridentBackendConfig에 adAdminUser를 지정합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 저장 클래스에 주석을 추가합니다.

   추가하세요 `trident.netapp.io/smbShareAdUser` 실패 없이 안전한 SMB를 활성화하기 위해 저장 클래스에 주석을 추가합니다. 주석에 지정된 사용자 값 `trident.netapp.io/smbShareAdUser` 사용자 이름과 동일해야 합니다. `smbcreds` 비밀. 다음 중 하나를 선택할 수 있습니다. `smbShareAdUserPermission` : `full_control` , `change` , 또는 `read` . 기본 권한은 다음과 같습니다. `full_control` .

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. PVC를 만듭니다.

   다음 예제에서는 PVC를 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS 구성 옵션 및 예

Trident 설치로 ONTAP NAS 드라이버를 만들고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClass에 매핑하기 위한 세부 정보를 제공합니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-nas`, `ontap-nas-economy` , 또는 `ontap-nas-flexgroup`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소, 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 원활한 MetroCluster 전환을 위해서는 다음을 참조하세요.MetroCluster 예제 .	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	기본
`dataLIF`	프로토콜 LIF의 IP 주소. NetApp 다음을 지정하는 것이 좋습니다. `dataLIF` . 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. 초기 설정 후 변경이 가능합니다. 참조하다 . IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 메트로클러스터는 생략합니다. 를 참조하십시오MetroCluster 예제 .	지정되지 않은 경우 지정된 주소 또는 SVM에서 파생됨(권장하지 않음)
`svm`	사용할 스토리지 가상 머신 **Metrocluster**의 경우 생략 를 참조하십시오MetroCluster 예제 .	SVM의 경우 파생됨 `managementLIF` 지정됨
`autoExportPolicy`	자동 내보내기 정책 생성 및 업데이트 활성화[부울] 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	거짓
`autoExportCIDRs`	Kubernetes 노드 IP를 필터링할 CIDR 목록 `autoExportPolicy` 활성화되어 있습니다. 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	["0.0.0.0/0", "::/0"]`
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됨	""
`username`	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	
`password`	클러스터/SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	

매개변수	설명	기본
`storagePrefix`	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 설정 후 업데이트 불가 ontap-nas-economy와 24자 이상의 storagePrefix를 사용하는 경우 볼륨 이름에는 스토리지 접두사가 포함되지만 qtree에는 포함되지 않습니다.	"삼지창"
`aggregate`	프로비저닝을 위한 집계(선택 사항, 설정된 경우 SVM에 할당해야 함). 를 위해 `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당되지 않은 경우 사용 가능한 모든 집계를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 집계가 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계를 구성한 경우, 집계의 이름이 변경되거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 SVM에 있는 집계로 변경하거나 집계를 완전히 제거해야 합니다.	""
`limitAggregateUsage`	사용량이 이 백분율을 초과하면 프로비저닝에 실패합니다. * Amazon FSx for ONTAP 에는 적용되지 않습니다*.	"" (기본적으로 적용되지 않음)
플렉스그룹집계목록	프로비저닝을 위한 집계 목록(선택 사항, 설정된 경우 SVM에 할당해야 함). SVM에 할당된 모든 집계는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다. **ontap-nas-flexgroup** 스토리지 드라이버에 대해 지원됩니다. SVM에서 집계 목록이 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 목록이 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계 목록을 구성한 경우, 집계 목록의 이름이 바뀌거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 집계 목록을 SVM에 있는 목록으로 변경하거나 집계 목록을 완전히 제거해야 합니다.	""

매개변수	설명	기본
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 qtree에 대해 관리하는 볼륨의 최대 크기를 제한합니다. qtreesPerFlexvol 옵션을 사용하면 FlexVol volume 당 최대 Qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true} 사용하지 마십시오. debugTraceFlags 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.	널
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 nfs , smb 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
nfsMountOptions	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 지속형 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident 는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스나 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident 연관된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
qtreesPerFlexvol	FlexVol 당 최대 Qtree는 [50, 300] 범위 내에 있어야 합니다.	"200"
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름, Trident SMB 공유를 만들 수 있도록 하는 이름, 또는 매개변수를 비워 두면 볼륨에 대한 일반 공유 액세스를 방지할 수 있습니다. 이 매개변수는 온프레미스 ONTAP 의 경우 선택 사항입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필수이므로 비워둘 수 없습니다.	smb-share
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. useREST`설정 시 `true , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. false Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. ontapi 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. vsadmin 그리고 cluster-admin 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 useREST 로 설정됩니다 true 기본적으로; 변경 useREST 에게 false ONTAPI(ZAPI) 호출을 사용합니다.	true`ONTAP 9.15.1 이상인 경우, 그렇지 않은 경우 `false.
limitVolumePoolSize	ontap-nas-economy 백엔드에서 Qtrees를 사용할 때 요청 가능한 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본
denyNewVolumePools	제한하다 ontap-nas-economy 백엔드가 Qtree를 포함하기 위해 새로운 FlexVol 볼륨을 생성하지 못하도록 합니다. 새로운 PV를 프로비저닝하는 데는 기존 Flexvol만 사용됩니다.	
adAdminUser	SMB 공유에 대한 전체 액세스 권한이 있는 Active Directory 관리자 사용자 또는 사용자 그룹입니다. 이 매개변수를 사용하면 SMB 공유에 대한 전체 제어 권한을 관리자에게 제공할 수 있습니다.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. defaults 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
spaceAllocation	Qtrees에 대한 공간 할당	"진실"
spaceReserve	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움)	"없음"
snapshotPolicy	사용할 스냅샷 정책	"없음"
qosPolicy	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하세요.	""
adaptiveQosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. ontap-nas-economy에서 지원되지 않습니다.	""
snapshotReserve	스냅샷을 위해 예약된 볼륨의 백분율	"0"이면 snapshotPolicy "없음"이고, 그렇지 않으면 ""
splitOnClone	생성 시 부모로부터 복제본을 분할합니다.	"거짓"
encryption	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. false . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	"거짓"
tieringPolicy	"없음"을 사용하는 계층화 정책	
unixPermissions	새로운 볼륨에 대한 모드	NFS 볼륨의 경우 "777", SMB 볼륨의 경우 비어 있음(해당 없음)
snapshotDir	에 대한 액세스를 제어합니다. .snapshot 예배 규칙서	NFSv4의 경우 "true", NFSv3의 경우 "false"

매개변수	설명	기본
`exportPolicy`	사용할 수출 정책	"기본"
`securityStyle`	새로운 볼륨에 대한 보안 스타일입니다. NFS 지원 `mixed` 그리고 `unix` 보안 스타일. SMB 지원 `mixed` 그리고 `ntfs` 보안 스타일.	NFS 기본값은 `unix` . SMB 기본값은 `ntfs` .
`nameTemplate`	사용자 정의 볼륨 이름을 만드는 템플릿입니다.	""

Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되는지 확인해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.

볼륨 프로비저닝 예시

기본값이 정의된 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

을 위한 `ontap-nas` 그리고 `ontap-nas-flexgroups` Trident 이제 FlexVol snapshotReserve 백분율과 PVC에 맞게 올바르게 조정되도록 새로운 계산을 사용합니다. 사용자가 PVC를 요청하면 Trident 새로운 계산을 사용하여 더 많은 공간을 가진 원래 FlexVol 생성합니다. 이 계산은 사용자가 PVC에서 요청한 쓰기 가능 공간을 확보하고 요청한 것보다 적은 공간을 확보하지 않도록 보장합니다. v21.07 이전에는 사용자가 스냅샷 예약 비율을 50%로 설정한 PVC(예: 5GiB)를 요청하면 2.5GiB의 쓰기 가능 공간만 확보되었습니다. 이는 사용자가 요청한 것이 전체 볼륨이기

때문입니다. `snapshotReserve` 그 중 일부입니다. Trident 21.07을 사용하면 사용자가 요청하는 것은 쓰기 가능한 공간이며 Trident 이를 정의합니다. `snapshotReserve` 전체 볼륨에 대한 백분율로 나타낸 숫자입니다. 이것은 적용되지 않습니다 `ontap-nas-economy` . 작동 방식을 알아보려면 다음 예를 참조하세요.

계산은 다음과 같습니다.

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

에서 요청한 5GiB입니다. 그만큼 `volume show` 명령을 실행하면 다음 예와 비슷한 결과가 표시됩니다.

이전 설치의 기존 백엔드는 Trident 업그레이드 시 위에서 설명한 대로 볼륨을 프로비저닝합니다. 업그레이드 전에 생성한 볼륨의 경우, 변경 사항을 적용하려면 볼륨 크기를 조정해야 합니다. 예를 들어, 2GiB PVC `snapshotReserve=50` 이전에는 1GiB의 쓰기 가능 공간을 제공하는 볼륨이 생성되었습니다. 예를 들어 볼륨 크기를 3GiB로 조정하면 애플리케이션은 6GiB 볼륨에서 3GiB의 쓰기 가능 공간을 확보하게 됩니다.

최소 구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

> Trident 와 함께 NetApp ONTAP 에서 Amazon FSx 사용하는 경우 IP 주소 대신 LIF에 DNS 이름을 지정하는 것이 좋습니다.

ONTAP NAS 경제 사례

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

ONTAP NAS Flexgroup 예시

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

MetroCluster 예제

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다."SVM 복제 및 복구" .

원활한 전환 및 스위치백을 위해 다음을 사용하여 SVM을 지정하십시오. `managementLIF` 그리고 생략하다 `dataLIF` 그리고 `svm` 매개변수. 예를 들어:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

SMB 볼륨 예시

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

인증서 기반 인증 예제

이는 최소 백엔드 구성의 예입니다. `clientCertificate` , `clientPrivateKey` , 그리고 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)이 채워집니다. `backend.json` 그리고 클라이언트 인증서, 개인 키, 신뢰할 수 있는 CA 인증서의 base64 인코딩 값을 각각 가져옵니다.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

자동 내보내기 정책 예

이 예제에서는 Trident 동적 내보내기 정책을 사용하여 내보내기 정책을 자동으로 만들고 관리하는 방법을 보여줍니다. 이것은 다음과 같은 경우에도 동일하게 작동합니다. `ontap-nas-economy` 그리고 `ontap-nas-flexgroup` 운전자.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

IPv6 주소 예

이 예에서는 다음을 보여줍니다. `managementLIF` IPv6 주소를 사용합니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

SMB 볼륨을 사용하는 Amazon FSx for ONTAP 예제

그만큼 `smbShare` SMB 볼륨을 사용하는 FSx for ONTAP 에는 매개변수가 필요합니다.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

가상 풀을 사용한 백엔드의 예

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 다음과 같은 특정 기본값이 설정됩니다. `spaceReserve` 없음, `spaceAllocation` 거짓이고, `encryption` 거짓. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident "주석" 필드에 프로비저닝 라벨을 설정합니다. FlexVol 에 대한 의견이 설정되었습니다. `ontap-nas` 또는 FlexGroup 용 `ontap-nas-flexgroup` . Trident 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이러한 예에서 일부 스토리지 풀은 자체적으로 설정합니다. `spaceReserve` , `spaceAllocation` , 그리고 `encryption` 값이 지정되고 일부 풀은 기본값을 재정의합니다.

ONTAP NAS 예시

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
```

```
      app: wordpress
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

ONTAP NAS FlexGroup 예시

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
```

```
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

ONTAP NAS 경제 사례

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

백엔드를 StorageClass에 매핑

다음 StorageClass 정의는 다음을 참조합니다.가상 풀을 사용한 백엔드의 예 . 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

- 그만큼 `protection-gold` StorageClass는 첫 번째 및 두 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 골드 레벨 보호를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 그만큼 `protection-not-gold` StorageClass는 세 번째 및 네 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 이것들은 금 이외의 보호 수준을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 그만큼 `app-mysqldb` StorageClass는 네 번째 가상 풀에 매핑됩니다. `ontap-nas` 백엔드. 이는 mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 그 `protection-silver-creditpoints-20k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 이 풀은 실버 레벨 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 그만큼 `creditpoints-5k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-nas` 백엔드와 두 번째 가상 풀 `ontap-nas-economy` 백엔드. 5000 크레딧 포인트를 제공하는 유일한 풀 서비스입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

업데이트 `dataLIF` 초기 구성 후

다음 명령을 실행하여 업데이트된 dataLIF로 새 백엔드 JSON 파일을 제공하면 초기 구성 후 dataLIF를 변경할 수 있습니다.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

PVC가 하나 이상의 포드에 연결된 경우 새로운 dataLIF가 적용되려면 해당 포드를 모두 내렸다가 다시 올려야 합니다.

보안 **SMB** 사례

ontap-nas 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas-economy 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

스토리지 풀을 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

추가했는지 확인하세요 `annotations` 안전한 SMB를 구현합니다. 보안 SMB는 백엔드나 PVC에 설정된 구성과 관계없이 주석 없이는 작동하지 않습니다.

ontap-nas-economy 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

단일 **AD** 사용자가 있는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

여러 **AD** 사용자가 있는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP 과 함께 **Trident** 사용

"Amazon FSx for NetApp ONTAP"NetApp ONTAP 스토리지 운영 체제를 기반으로 하는 파일 시스템을 실행하고 실행할 수 있는 완전 관리형 AWS 서비스입니다. FSx for ONTAP 사용하면 AWS에 데이터를 저장함으로써 얻는 단순성, 민첩성, 보안성, 확장성의 이점을 누리는 동시에, 익숙한 NetApp 기능, 성능, 관리 역량을 활용할 수 있습니다. FSx for ONTAP ONTAP 파일 시스템 기능과 관리 API를 지원합니다.

Amazon FSx for NetApp ONTAP 파일 시스템을 Trident 와 통합하면 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터가 ONTAP 에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있습니다.

파일 시스템은 온프레미스 ONTAP 클러스터와 유사한 Amazon FSx 의 기본 리소스입니다. 각 SVM 내에서 하나 이상의 볼륨을 만들 수 있습니다. 볼륨은 파일 시스템의 파일과 폴더를 저장하는 데이터 컨테이너입니다. Amazon FSx for NetApp ONTAP 클라우드에서 관리형 파일 시스템으로 제공됩니다. 새로운 파일 시스템 유형은 * NetApp ONTAP*이라고 합니다.

Amazon FSx for NetApp ONTAP 과 함께 Trident 사용하면 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터가 ONTAP 에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있습니다.

요구 사항

또한"Trident 요구 사항" FSx for ONTAP Trident 와 통합하려면 다음이 필요합니다.

- 기존 Amazon EKS 클러스터 또는 자체 관리 Kubernetes 클러스터 `kubectl` 설치됨.
- 클러스터의 워커 노드에서 접근할 수 있는 기존 Amazon FSx for NetApp ONTAP 파일 시스템 및 스토리지 가상 머신(SVM)입니다.
- 준비된 작업자 노드"NFS 또는 iSCSI" .

> Amazon Linux 및 Ubuntu에 필요한 노드 준비 단계를 따르세요. "Amazon Machine Images" (AMI)는 EKS AMI 유형에 따라 다릅니다.

고려 사항

- SMB 볼륨:
  - SMB 볼륨은 다음을 사용하여 지원됩니다. `ontap-nas` 운전자만.
  - SMB 볼륨은 Trident EKS 애드온에서 지원되지 않습니다.
  - Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 참조하다 "SMB 볼륨 프로비저닝 준비" 자세한 내용은.
- Trident 24.02 이전에는 자동 백업이 활성화된 Amazon FSx 파일 시스템에서 생성된 볼륨은 Trident 에서 삭제할 수 없었습니다. Trident 24.02 이상에서 이 문제를 방지하려면 다음을 지정하십시오. `fsxFilesystemID` , AWS `apiRegion` , AWS `apikey` , 그리고 AWS `secretKey` AWS FSx for ONTAP 의 백엔드 구성 파일에서.

> Trident 에 IAM 역할을 지정하는 경우 다음을 지정하지 않아도 됩니다. `apiRegion` , `apiKey` , 그리고 `secretKey` 필드를 Trident 에 명시적으로 지정합니다. 자세한 내용은 다음을 참조하세요 ."FSx for ONTAP 구성 옵션 및 예제" .

Trident SAN/iSCSI 및 **EBS-CSI** 드라이버 동시 사용

AWS(EKS, ROSA, EC2 또는 기타 인스턴스)에서 ontap-san 드라이버(예: iSCSI)를 사용하려는 경우 노드에 필요한 다중 경로 구성이 Amazon Elastic Block Store(EBS) CSI 드라이버와 충돌할 수 있습니다. 동일한 노드에 있는 EBS 디스크를 방해하지 않고 멀티패스 기능을 보장하려면 멀티패스 설정에서 EBS를 제외해야 합니다. 이 예에서는 다음을 보여줍니다. `multipath.conf` EBS 디스크를 다중 경로에서 제외하면서 필수 Trident 설정을 포함하는 파일:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

입증

Trident 두 가지 인증 모드를 제공합니다.

- 자격 증명 기반(권장): AWS Secrets Manager에 자격 증명을 안전하게 저장합니다. 당신은 사용할 수 있습니다 `fsxadmin` 파일 시스템 또는 사용자 `vsadmin` SVM에 맞게 사용자가 구성했습니다.

> Trident 다음과 같이 실행될 것으로 예상됩니다. `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자로 지정할 수 있습니다. Amazon FSx for NetApp ONTAP 다음이 있습니다. `fsxadmin` ONTAP 의 제한된 대체품인 사용자 `admin` 클러스터 사용자. 우리는 강력히 사용을 권장합니다 `vsadmin` Trident 와 함께.

- 인증서 기반: Trident SVM에 설치된 인증서를 사용하여 FSx 파일 시스템의 SVM과 통신합니다.

인증 활성화에 대한 자세한 내용은 드라이버 유형에 대한 인증을 참조하세요.

- "ONTAP NAS 인증"
- "ONTAP SAN 인증"

테스트된 **Amazon Machine Images(AMI)**

EKS 클러스터는 다양한 운영 체제를 지원하지만 AWS는 컨테이너와 EKS에 맞게 특정 Amazon Machine Image(AMI)를 최적화했습니다. 다음 AMI는 NetApp Trident 25.02에서 테스트되었습니다.

아미	NAS	NAS-경제	iSCSI	iSCSI 경제
AL2023_x86_64_STANDARD	예	예	예	예
AL2_x86_64	예	예	예*	예*
BOTTLEROCKET_x86_64	예**	예	해당 없음	해당 없음
AL2023_ARM_64_STANDARD	예	예	예	예
AL2_ARM_64	예	예	예*	예*
BOTTLEROCKET_ARM_64	예**	예	해당 없음	해당 없음

- * 노드를 재시작하지 않고는 PV를 삭제할 수 없습니다.
- ** Trident 버전 25.02에서는 NFSv3와 작동하지 않습니다.

> 원하는 AMI가 여기에 나열되어 있지 않더라도 지원되지 않는다는 의미는 아닙니다. 단순히 테스트를 거치지 않았다는 의미일 뿐입니다. 이 목록은 AMI가 작동하는 것으로 알려진 방법에 대한 가이드 역할을 합니다.

다음과 함께 수행된 테스트**:**

- EKS 버전: 1.32
- 설치 방법: Helm 25.06 및 AWS 추가 기능 25.06
- NAS의 경우 NFSv3와 NFSv4.1이 모두 테스트되었습니다.
- SAN의 경우 iSCSI만 테스트되었으며 NVMe-oF는 테스트되지 않았습니다.

수행된 테스트:

- 생성: 스토리지 클래스, pvc, pod
- 삭제: pod, pvc(일반, qtree/lun – 경제형, AWS 백업이 있는 NAS)

더 많은 정보를 찾아보세요

- "Amazon FSx for NetApp ONTAP 설명서"
- "Amazon FSx for NetApp ONTAP 에 대한 블로그 게시물"

IAM 역할 및 **AWS Secret** 생성

명시적인 AWS 자격 증명을 제공하는 대신 AWS IAM 역할로 인증하여 Kubernetes 포드가 AWS 리소스에 액세스하도록 구성할 수 있습니다.

> AWS IAM 역할을 사용하여 인증하려면 EKS를 사용하여 배포된 Kubernetes 클러스터가 있어야 합니다.

AWS Secrets Manager 비밀 만들기

Trident 스토리지를 관리하기 위해 FSx vserver에 대한 API를 발행하므로 이를 위해서는 자격 증명이 필요합니다. 이러한 자격 증명을 전달하는 안전한 방법은 AWS Secrets Manager 비밀을 사용하는 것입니다. 따라서 아직 없으면 vsadmin 계정의 자격 증명이 포함된 AWS Secrets Manager 비밀을 만들어야 합니다.

이 예제에서는 Trident CSI 자격 증명을 저장하기 위해 AWS Secrets Manager 비밀을 만듭니다.

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

IAM 정책 생성

Trident 도 올바르게 실행하려면 AWS 권한이 필요합니다. 따라서 Trident 필요한 권한을 부여하는 정책을 만들어야 합니다.

다음 예제에서는 AWS CLI를 사용하여 IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

정책 **JSON** 예시:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

서비스 계정 연결(IRSA)을 위한 Pod ID 또는 IAM 역할 생성

Kubernetes 서비스 계정을 구성하여 EKS Pod Identity 또는 서비스 계정 연결(IRSA)을 위한 IAM 역할을 통해 AWS Identity and Access Management(IAM) 역할을 맡을 수 있습니다. 서비스 계정을 사용하도록 구성된 모든 Pod는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

포드 아이덴티티

Amazon EKS Pod Identity 연결은 Amazon EC2 인스턴스 프로필이 Amazon EC2 인스턴스에 자격 증명을 제공하는 방식과 유사하게 애플리케이션의 자격 증명을 관리하는 기능을 제공합니다.

EKS 클러스터에 **Pod Identity** 설치:

AWS 콘솔을 통해 또는 다음 AWS CLI 명령을 사용하여 Pod ID를 생성할 수 있습니다.

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

자세한 내용은 다음을 참조하세요."Amazon EKS Pod Identity Agent 설정" .

trust-relationship.json 생성:

EKS 서비스 주체가 Pod Identity에 대한 이 역할을 수행할 수 있도록 trust-relationship.json을 생성합니다. 그런 다음 이 신뢰 정책으로 역할을 만듭니다.

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json 파일:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

IAM 역할에 역할 정책 첨부:

이전 단계의 역할 정책을 생성된 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

포드 **ID** 연결 생성:

IAM 역할과 Trident 서비스 계정(trident-controller) 간에 Pod ID 연결을 만듭니다.

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

서비스 계정 연결**(IRSA)**을 위한 **IAM** 역할

AWS CLI 사용:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json 파일**:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

다음 값을 업데이트하세요. `trust-relationship.json` 파일:

- **<account_id>** - AWS 계정 ID
- **<oidc_provider>** - EKS 클러스터의 OIDC입니다. 다음을 실행하여 oidc_provider를 얻을 수 있습니다.

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

IAM 정책을 사용하여 **IAM** 역할 연결:

역할이 생성되면 다음 명령을 사용하여 위 단계에서 생성된 정책을 역할에 연결합니다.

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

OICD 공급자가 연결되어 있는지 확인하세요:

OIDC 공급자가 클러스터와 연결되어 있는지 확인하세요. 다음 명령을 사용하여 확인할 수 있습니다.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

출력이 비어 있으면 다음 명령을 사용하여 IAM OIDC를 클러스터에 연결합니다.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

eksctl을 사용하는 경우 다음 예를 사용하여 EKS의 서비스 계정에 대한 IAM 역할을 생성하세요.

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Trident 설치

Trident Kubernetes에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다.

다음 방법 중 하나를 사용하여 Trident 설치할 수 있습니다.

- 지배
- EKS 애드온

스냅샷 기능을 활용하려면 CSI 스냅샷 컨트롤러 애드온을 설치하세요. 참조하다"CSI 볼륨에 대한 스냅샷 기능 활성화" 자세한 내용은.

helm을 통해 **Trident** 설치

포드 아이덴티티

1. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

2. 다음 예를 사용하여 Trident 설치하세요.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace
```

당신은 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등의 설치 세부 정보를 검토하는 명령입니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

서비스 계정 연결**(IRSA)**

1. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

2. 클라우드 공급자 및 *클라우드 ID*에 대한 값을 설정합니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

당신은 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등의 설치 세부 정보를 검토하는 명령입니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2506.0    25.06.0
```

iSCSI를 사용하려면 클라이언트 컴퓨터에서 iSCSI가 활성화되어 있는지 확인하세요. AL2023 Worker 노드 OS를 사용하는 경우 helm 설치에서 node prep 매개변수를 추가하여 iSCSI 클라이언트 설치를 자동화할 수 있습니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

EKS 애드온을 통해 Trident 설치

Trident EKS 애드온에는 최신 보안 패치와 버그 수정이 포함되어 있으며, AWS에서 Amazon EKS와 함께 작동하도록 검증되었습니다. EKS 추가 기능을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장할 수 있으며 추가 기능을 설치, 구성, 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

필수 조건

AWS EKS에 대한 Trident 추가 기능을 구성하기 전에 다음 사항이 있는지 확인하세요.

- 추가 구독이 있는 Amazon EKS 클러스터 계정
- AWS 마켓플레이스에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

AWS에 Trident 애드온 활성화

관리 콘솔

1. Amazon EKS 콘솔을 엽니다. https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 구성하려는 클러스터의 이름을 선택합니다.
4. *추가 기능*을 선택한 다음 *더 많은 추가 기능 받기*를 선택하세요.
5. 추가 기능을 선택하려면 다음 단계를 따르세요.
   a. **AWS Marketplace** 추가 기능 섹션까지 아래로 스크롤하여 검색 상자에 *"Trident"*를 입력합니다.
   b. NetApp 의 Trident 상자 오른쪽 상단에 있는 확인란을 선택하세요.
   c. *다음*을 선택하세요.
6. 선택한 추가 기능 구성 설정 페이지에서 다음을 수행합니다.

   Pod Identity 연결을 사용하는 경우 이 단계를 건너뜁니다.

   a. 사용하고 싶은 *버전*을 선택하세요.
   b. IRSA 인증을 사용하는 경우 선택적 구성 설정에서 사용 가능한 구성 값을 설정해야 합니다.
      - 사용하고 싶은 *버전*을 선택하세요.
      - 추가 기능 구성 스키마*를 따르고 *구성 값 섹션의 **configurationValues** 매개변수를 이전 단계에서 만든 role-arn으로 설정합니다(값은 다음 형식이어야 함).

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
충돌 해결 방법에 대해 재정의를 선택하면 기존 추가 기능에 대한 하나 이상의 설정을 Amazon EKS 추가 기능 설정으로 덮어쓸 수 있습니다. 이 옵션을 활성화하지 않고 기존 설정과 충돌이 발생하면 작업이 실패합니다. 발생한 오류 메시지를 사용하여 충돌 문제를 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 사용자가 직접 관리해야 하는 설정을 관리하지 않는지 확인하세요.

7. *다음*을 선택하세요.
8. 검토 및 추가 페이지에서 *만들기*를 선택하세요.

   애드온 설치가 완료되면 설치된 애드온이 표시됩니다.

AWS CLI

1. 생성하다 **`add-on.json`** 파일:

Pod Identity의 경우 다음 형식을 사용하세요:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

IRSA 인증의 경우 다음 형식을 사용하세요:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

바꾸다 `<role ARN>` 이전 단계에서 생성된 역할의 ARN을 사용합니다.

2. Trident EKS 애드온을 설치하세요.

```
aws eks create-addon --cli-input-json file://add-on.json
```

엑시틀

다음 예제 명령은 Trident EKS 추가 기능을 설치합니다.

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Trident EKS 애드온 업데이트

관리 콘솔

1. Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 업데이트하려는 클러스터의 이름을 선택합니다.
4. 추가 기능 탭을 선택하세요.
5. * NetApp 의 Trident *를 선택한 다음 *편집*을 선택합니다.
6. * NetApp 의 Trident 구성* 페이지에서 다음을 수행합니다.
   a. 사용하고 싶은 *버전*을 선택하세요.
   b. *선택적 구성 설정*을 확장하고 필요에 따라 수정합니다.
   c. *변경 사항 저장*을 선택하세요.

AWS CLI

다음 예제에서는 EKS 추가 기능을 업데이트합니다.

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

엑시틀

- FSxN Trident CSI 애드온의 현재 버전을 확인하세요. 바꾸다 `my-cluster` 클러스터 이름으로.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  예시 출력:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 이전 단계의 출력에서 UPDATE AVAILABLE에 반환된 버전으로 애드온을 업데이트합니다.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

제거하면 `--force` 옵션과 Amazon EKS 추가 기능 설정이 기존 설정과 충돌하는 경우 Amazon EKS 추가 기능 업데이트가 실패합니다. 충돌을 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 사용자가 관리해야 하는 설정을 관리하지 않는지 확인하세요. 이 옵션을 사용하면 해당 설정이 덮어쓰기되기 때문입니다. 이 설정에 대한 다른 옵션에 대한 자세한 내용은 다음을 참조하세요."애드온" . Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 다음을 참조하세요."쿠버네티스 필드 관리" .

Trident EKS 애드온 제거/제거

Amazon EKS 추가 기능을 제거하는 데는 두 가지 옵션이 있습니다.

- 클러스터에 추가 소프트웨어 유지 – 이 옵션을 선택하면 Amazon EKS에서 모든 설정을 관리하지 않습니다. 또한 Amazon EKS에서 업데이트를 알리고, 업데이트를 시작한 후 Amazon EKS 추가 기능을 자동으로 업데이트하는 기능도 제거됩니다. 하지만 클러스터의 추가 소프트웨어는 그대로 유지됩니다. 이 옵션을 선택하면 추가 기능이 Amazon EKS 추가 기능이 아닌 자체 관리형 설치가 됩니다. 이 옵션을 사용하면 추가 기능에 다운타임이 발생하지 않습니다. 유지하다 `--preserve` 추가 기능을 보존하려면 명령에 옵션을 추가하세요.
- 클러스터에서 애드온 소프트웨어를 완전히 제거합니다 – NetApp 클러스터에 종속된 리소스가 없는 경우에만 클러스터에서 Amazon EKS 애드온을 제거할 것을 권장합니다. 제거하다 `--preserve` 옵션에서 `delete` 추가 기능을 제거하는 명령입니다.

추가 기능에 IAM 계정이 연결되어 있는 경우 해당 IAM 계정은 제거되지 않습니다.

관리 콘솔

1. Amazon EKS 콘솔을 엽니다. https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 제거할 클러스터의 이름을 선택합니다.
4. 추가 기능 탭을 선택한 다음 * Trident by NetApp*을 선택합니다.*
5. *제거*를 선택하세요.
6. **netapp_trident-operator** 제거 확인 대화 상자에서 다음을 수행합니다.
   a. Amazon EKS가 애드온에 대한 설정 관리를 중지하도록 하려면 *클러스터에 유지*를 선택합니다. 클러스터에 애드온 소프트웨어를 유지하여 애드온의 모든 설정을 직접 관리하려는 경우 이렇게 하세요.
   b. *netapp_trident-operator*를 입력합니다.
   c. *제거*를 선택하세요.

AWS CLI

바꾸다 `my-cluster` 클러스터 이름을 입력한 후 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

엑시틀

다음 명령은 Trident EKS 애드온을 제거합니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

스토리지 백엔드 구성

ONTAP SAN 및 NAS 드라이버 통합

스토리지 백엔드를 만들려면 JSON 또는 YAML 형식으로 구성 파일을 만들어야 합니다. 이 파일에는 원하는 저장 유형(NAS 또는 SAN), 파일 시스템, 데이터를 가져올 SVM, 그리고 이를 통해 인증하는 방법을 지정해야 합니다. 다음 예제에서는 NAS 기반 스토리지를 정의하고 AWS 비밀번호를 사용하여 사용하려는 SVM에 자격 증명을 저장하는 방법을 보여줍니다.

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

다음 명령을 실행하여 Trident 백엔드 구성(TBC)을 만들고 검증합니다.

- yaml 파일에서 트라이던트 백엔드 구성(TBC)을 만들고 다음 명령을 실행합니다.

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- 트라이던트 백엔드 구성(TBC)이 성공적으로 생성되었는지 확인합니다.

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE    STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9    Bound    Success
```

FSx for ONTAP 드라이버 세부 정보

다음 드라이버를 사용하여 Trident Amazon FSx for NetApp ONTAP 과 통합할 수 있습니다.

- `ontap-san`: 프로비저닝된 각 PV는 자체 Amazon FSx for NetApp ONTAP 볼륨 내의 LUN입니다. 블록 저장에 권장됩니다.
- `ontap-nas`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 입니다. NFS 및 SMB에 권장됩니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 LUN 수가 있는 LUN입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 qtree이며, Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 qtree 수가 있습니다.
- `ontap-nas-flexgroup`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP FlexGroup 볼륨을 위한 전체 Amazon FSx입니다.

운전자 세부 정보는 다음을 참조하세요."NAS 드라이버" 그리고"SAN 드라이버" .

구성 파일이 생성되면 다음 명령을 실행하여 EKS 내에서 해당 파일을 생성합니다.

```
kubectl create -f configuration_file
```

상태를 확인하려면 다음 명령을 실행하세요.

```
kubectl get tbc -n trident
```

```
NAME                      BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

백엔드 고급 구성 및 예제

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	예
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-nas`, `ontap-nas-economy` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소, 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다. 당신이 제공하는 경우 `fsxFilesystemID` 아래에 `aws` 필드에서는 다음을 제공할 필요가 없습니다. `managementLIF` Trident SVM을 검색하기 때문에 `managementLIF` AWS에서 제공하는 정보입니다. 따라서 SVM(예: vsadmin)에서 사용자에 대한 자격 증명을 제공해야 하며 사용자에게 다음이 있어야 합니다. `vsadmin` 역할.	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	예
`dataLIF`	프로토콜 LIF의 IP 주소. * ONTAP NAS 드라이버*: NetApp dataLIF를 지정하는 것을 권장합니다. 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. 초기 설정 후 변경이 가능합니다. 참조하다 . * ONTAP SAN 드라이버*: iSCSI에 대해서는 지정하지 마세요. Trident ONTAP Selective LUN Map을 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCI LIF를 검색합니다. dataLIF가 명시적으로 정의된 경우 경고가 생성됩니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다.	
`autoExportPolicy`	자동 내보내기 정책 생성 및 업데이트 활성화[부울] 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	`false`
`autoExportCIDRs`	Kubernetes 노드 IP를 필터링할 CIDR 목록 `autoExportPolicy` 활성화되어 있습니다. 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	"["0.0.0.0/0", "::/0"]"
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됩니다.	""

매개변수	설명	예
`username`	클러스터 또는 SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. 예를 들어, vsadmin.	
`password`	클러스터 또는 SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다.	
`svm`	사용할 스토리지 가상 머신	SVM managementLIF가 지정된 경우 파생됩니다.
`storagePrefix`	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 생성 후에는 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새로운 백엔드를 만들어야 합니다.	`trident`
`limitAggregateUsage`	* Amazon FSx for NetApp ONTAP에 대해서는 지정하지 마세요.* 제공된 `fsxadmin` 그리고 `vsadmin` Trident 사용하여 집계 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다.	사용하지 마세요.
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 qtree 및 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다. `qtreesPerFlexvol` 옵션을 사용하면 FlexVol volume 당 최대 Qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	Flexvol 볼륨당 최대 LUN은 [50, 200] 범위 내에 있어야 합니다. SAN만 해당.	“100”
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true} 사용하지 마십시오. `debugTraceFlags` 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.	널
`nfsMountOptions`	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 지속형 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident 는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스나 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident 연관된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""

매개변수	설명	예
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 nfs , smb , 또는 null. 설정해야 합니다 **smb SMB** 볼륨의 경우. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
qtreesPerFlexvol	FlexVol volume 당 최대 Qtree는 [50, 300] 범위 내에 있어야 합니다.	"200"
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름 또는 Trident SMB 공유를 만들 수 있도록 하는 이름입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필요합니다.	smb-share
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. 설정 시 true Trident ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. ontap 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. vsadmin 그리고 cluster-admin 역할.	false
aws	AWS FSx for ONTAP 의 구성 파일에서 다음을 지정할 수 있습니다. fsxFilesystemID : AWS FSx 파일 시스템의 ID를 지정합니다. - apiRegion : AWS API 지역 이름. - apikey : AWS API 키. - secretKey : AWS 비밀 키.	"" "" ""
credentials	AWS Secrets Manager에 저장할 FSx SVM 자격 증명을 지정합니다. - name : SVM의 자격 증명을 포함하는 비밀의 Amazon 리소스 이름(ARN)입니다. - type : 설정 awsarn . 참조하다"AWS Secrets Manager 비밀 만들기" 자세한 내용은.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. defaults 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
`spaceAllocation`	LUN에 대한 공간 할당	`true`
`spaceReserve`	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움)	`none`
`snapshotPolicy`	사용할 스냅샷 정책	`none`
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용해야 하며, 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. ontap-nas-economy에서 지원되지 않습니다.	""
`snapshotReserve`	스냅샷에 예약된 볼륨의 백분율 "0"	만약에 `snapshotPolicy` ~이다 `none` , `else` ""
`splitOnClone`	생성 시 부모로부터 복제본을 분할합니다.	`false`
`encryption`	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. `false` . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	`false`
`luksEncryption`	LUKS 암호화를 활성화합니다. 참조하다"Linux Unified Key Setup(LUKS) 사용" . SAN만 해당.	""
`tieringPolicy`	사용할 계층화 정책 `none`	
`unixPermissions`	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워 두세요.	""

매개변수	설명	기본
`securityStyle`	새로운 볼륨에 대한 보안 스타일입니다. NFS 지원 `mixed` 그리고 `unix` 보안 스타일. SMB 지원 `mixed` 그리고 `ntfs` 보안 스타일.	NFS 기본값은 `unix` . SMB 기본값은 `ntfs` .

SMB 볼륨 프로비저닝 준비

다음을 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. `ontap-nas` 운전사. 완료하기 전에ONTAP SAN 및 NAS 드라이버 통합 다음 단계를 완료하세요.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음을 수행하십시오. `ontap-nas` 운전자는 다음 사항을 갖춰야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2019를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명을 포함하는 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

단계

1. SMB 주식을 생성합니다. 다음 두 가지 방법 중 하나를 사용하여 SMB 관리자 공유를 생성할 수 있습니다 ."Microsoft 관리 콘솔" 공유 폴더 스냅인 또는 ONTAP CLI 사용. ONTAP CLI를 사용하여 SMB 공유를 생성하려면:
   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 만듭니다.

      그만큼 `vserver cifs share create` 이 명령은 공유 생성 중에 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연관된 SMB 공유를 만듭니다.

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. 공유가 생성되었는지 확인하세요.

```
vserver cifs share show -share-name share_name
```

      참조하다"SMB 공유 만들기" 자세한 내용은 다음을 참조하세요.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대해서는 다음을 참조하세요."FSx for ONTAP 구성 옵션 및 예제" .

매개변수	설명	예
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름 또는 Trident SMB 공유를 만들 수 있도록 하는 이름입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필요합니다.	`smb-share`
`nasType`	설정해야 합니다 **`smb`** . null인 경우 기본값은 다음과 같습니다. `nfs` .	`smb`
`securityStyle`	새로운 볼륨에 대한 보안 스타일입니다. 설정해야 합니다 **`ntfs`** 또는 **`mixed`** **SMB** 볼륨의 경우.	`ntfs`또는 `mixed` SMB 볼륨의 경우
`unixPermissions`	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워두어야 합니다.	""

스토리지 클래스 및 **PVC** 구성

Kubernetes StorageClass 객체를 구성하고 Trident 가 볼륨을 프로비저닝하는 방법을 지시하는 스토리지 클래스를 만듭니다. 구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim(PVC)을 만듭니다. 그런 다음 PV를 포드에 장착할 수 있습니다.

스토리지 클래스 생성

Kubernetes StorageClass 객체 구성

그만큼 "Kubernetes StorageClass 객체" 객체는 Trident 해당 클래스에 사용되는 프로비저너로 식별하고 Trident 볼륨을 프로비저닝하는 방법을 지시합니다. NFS를 사용하여 볼륨에 대한 Storageclass를 설정하려면 이 예제를 사용하세요(전체 속성 목록은 아래의 Trident 속성 섹션을 참조하세요).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

iSCSI를 사용하여 볼륨에 대한 Storageclass를 설정하려면 다음 예를 사용하세요.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

AWS Bottlerocket에서 NFSv3 볼륨을 프로비저닝하려면 필요한 항목을 추가하세요. `mountOptions` 저장 클래스로:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

스토리지 클래스 생성

단계

1. 이것은 Kubernetes 객체이므로 다음을 사용합니다. `kubectl` Kubernetes에서 생성하세요.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스를 볼 수 있어야 하며, Trident 가 백엔드에서 풀을 검색했어야 합니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

PVC를 만듭니다

에이 "지속적 볼륨 클레임" (PVC)는 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

PVC는 특정 크기 또는 액세스 모드의 저장소를 요청하도록 구성될 수 있습니다. 연관된 StorageClass를 사용하면 클러스터 관리자는 PersistentVolume 크기 및 액세스 모드(성능이나 서비스 수준 등) 이상을 제어할 수 있습니다.

PVC를 만든 후에는 볼륨을 포드에 마운트할 수 있습니다.

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

다음 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWX 접근이 가능한 **PVC**

이 예에서는 StorageClass와 연관된 RWX 액세스가 있는 기본 PVC를 보여줍니다. `basic-csi` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

iSCSI를 사용한 **PVC** 예제

이 예에서는 RWO 액세스가 있는 iSCSI용 기본 PVC를 보여줍니다. 이 PVC는 StorageClass와 연관되어 있습니다. `protection-gold` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 1Gi
storageClassName: protection-gold
```

PVC 생성

단계

1. PVC을 생성합니다.

```
kubectl create -f pvc.yaml
```

2. PVC 상태를 확인하세요.

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi       RWO                          5m
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

Trident 속성

이러한 매개변수는 주어진 유형의 볼륨을 프로비저닝하는 데 어떤 Trident 관리 스토리지 풀을 사용해야 하는지 결정합니다.

기인하다	유형	가치	권하다	요구	지원됨
미디어[1]	끈	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다 의미합니다.	지정된 미디어 유형	온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-산, 솔리드파이어-산
프로비저닝 유형	끈	얇은, 두꺼운	풀은 이 프로비저닝 방법을 지원합니다.	프로비저닝 방법이 지정됨	두꺼운: 모두 온탭; 얇은: 모두 온탭 & solidfire-san
백엔드 유형	끈	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	풀은 이 유형의 백엔드에 속합니다.	백엔드 지정됨	모든 운전자
스냅샷	부울	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다.	스냅샷이 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
클론	부울	참, 거짓	풀은 볼륨 복제를 지원합니다.	복제가 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
암호화	부울	참, 거짓	풀은 암호화된 볼륨을 지원합니다.	암호화가 활성화된 볼륨	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산

기인하다	유형	가치	권하다	요구	지원됨
아이옵스	정수	양의 정수	풀은 이 범위에서 IOPS를 보장할 수 있습니다.	볼륨은 이러한 IOPS를 보장합니다.	솔리드파이어-산

[1]: ONTAP Select 시스템에서는 지원되지 않습니다.

샘플 애플리케이션 배포

스토리지 클래스와 PVC가 생성되면 PV를 포드에 마운트할 수 있습니다. 이 섹션에서는 PV를 포드에 연결하는 명령과 구성의 예를 나열합니다.

단계

1. 볼륨을 포드에 마운트합니다.

```
kubectl create -f pv-pod.yaml
```

다음 예에서는 PVC를 포드에 부착하기 위한 기본 구성을 보여줍니다. 기본 구성:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

다음을 사용하여 진행 상황을 모니터링할 수 있습니다. `kubectl get pod --watch` .

2. 볼륨이 마운트되었는지 확인하세요. `/my/mount/path` .

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                                 Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

이제 Pod를 삭제할 수 있습니다. Pod 애플리케이션은 더 이상 존재하지 않지만 볼륨은 그대로 유지됩니다.

```
kubectl delete pod pv-pod
```

EKS 클러스터에서 **Trident EKS** 추가 기능 구성

NetApp Trident Kubernetes에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다. NetApp Trident EKS 애드온에는 최신 보안 패치와 버그 수정이 포함되어 있으며, AWS에서 Amazon EKS와 함께 작동하도록 검증되었습니다. EKS 추가 기능을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장할 수 있으며 추가 기능을 설치, 구성, 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

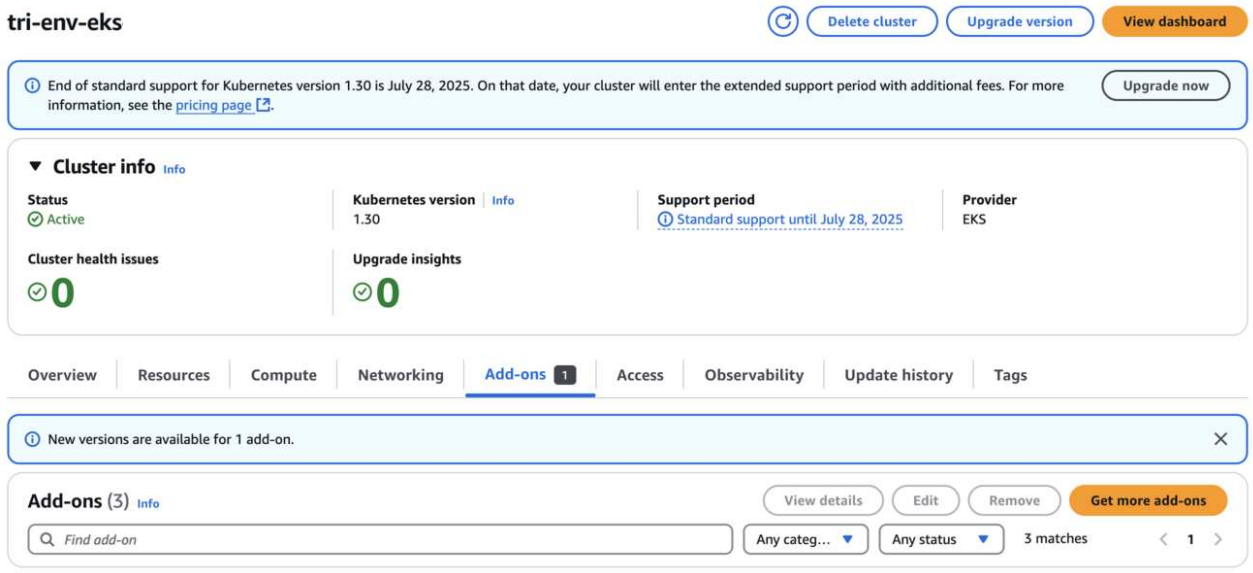
필수 조건

AWS EKS에 대한 Trident 추가 기능을 구성하기 전에 다음 사항이 있는지 확인하세요.

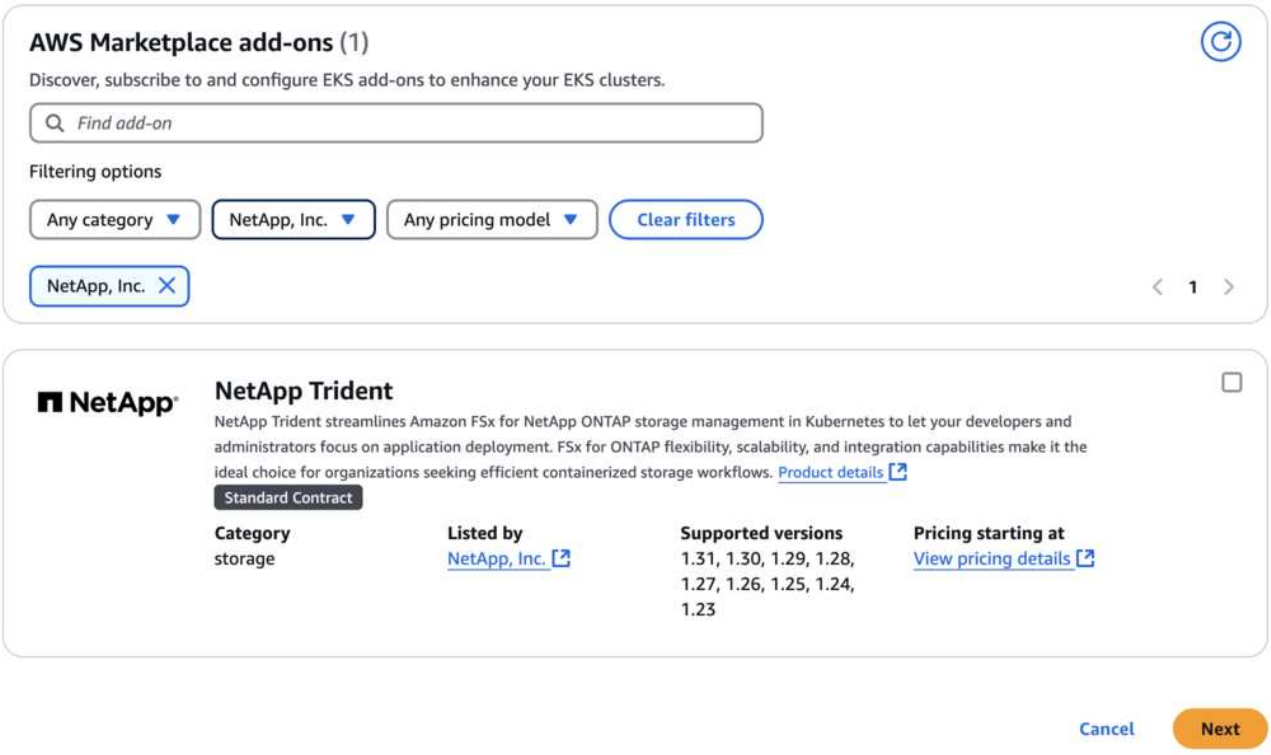
- 애드온을 사용할 수 있는 권한이 있는 Amazon EKS 클러스터 계정입니다. 참조하다"Amazon EKS 애드온" .
- AWS 마켓플레이스에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

단계

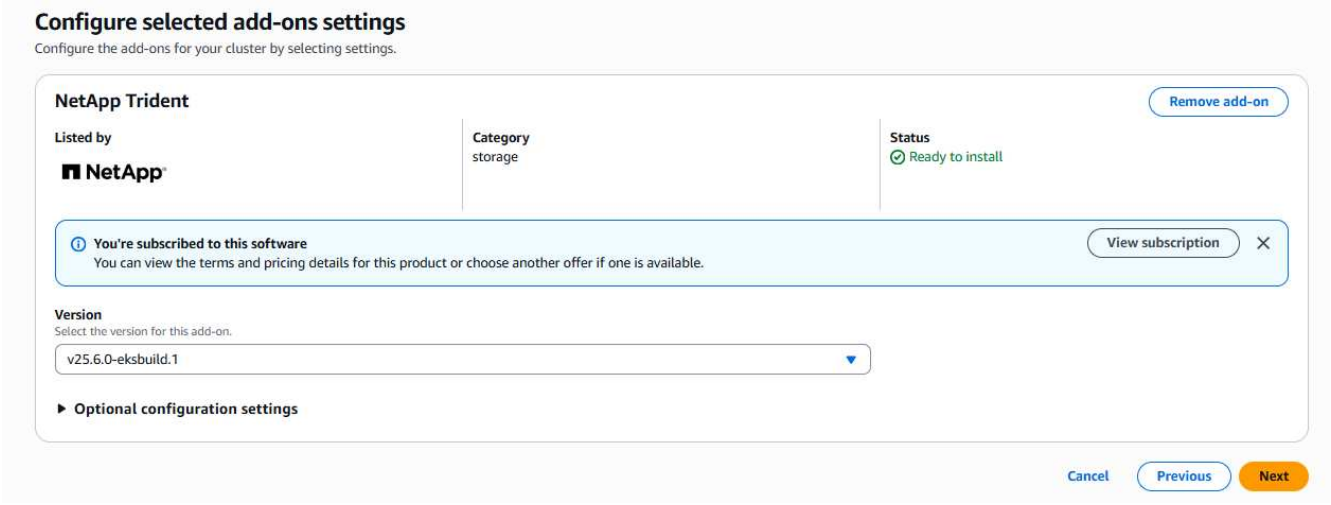
1. EKS Pod가 AWS 리소스에 액세스할 수 있도록 IAM 역할과 AWS 비밀번호를 생성해야 합니다. 지침은 다음을 참조하세요."IAM 역할 및 AWS Secret 생성" .
2. EKS Kubernetes 클러스터에서 추가 기능 탭으로 이동합니다.



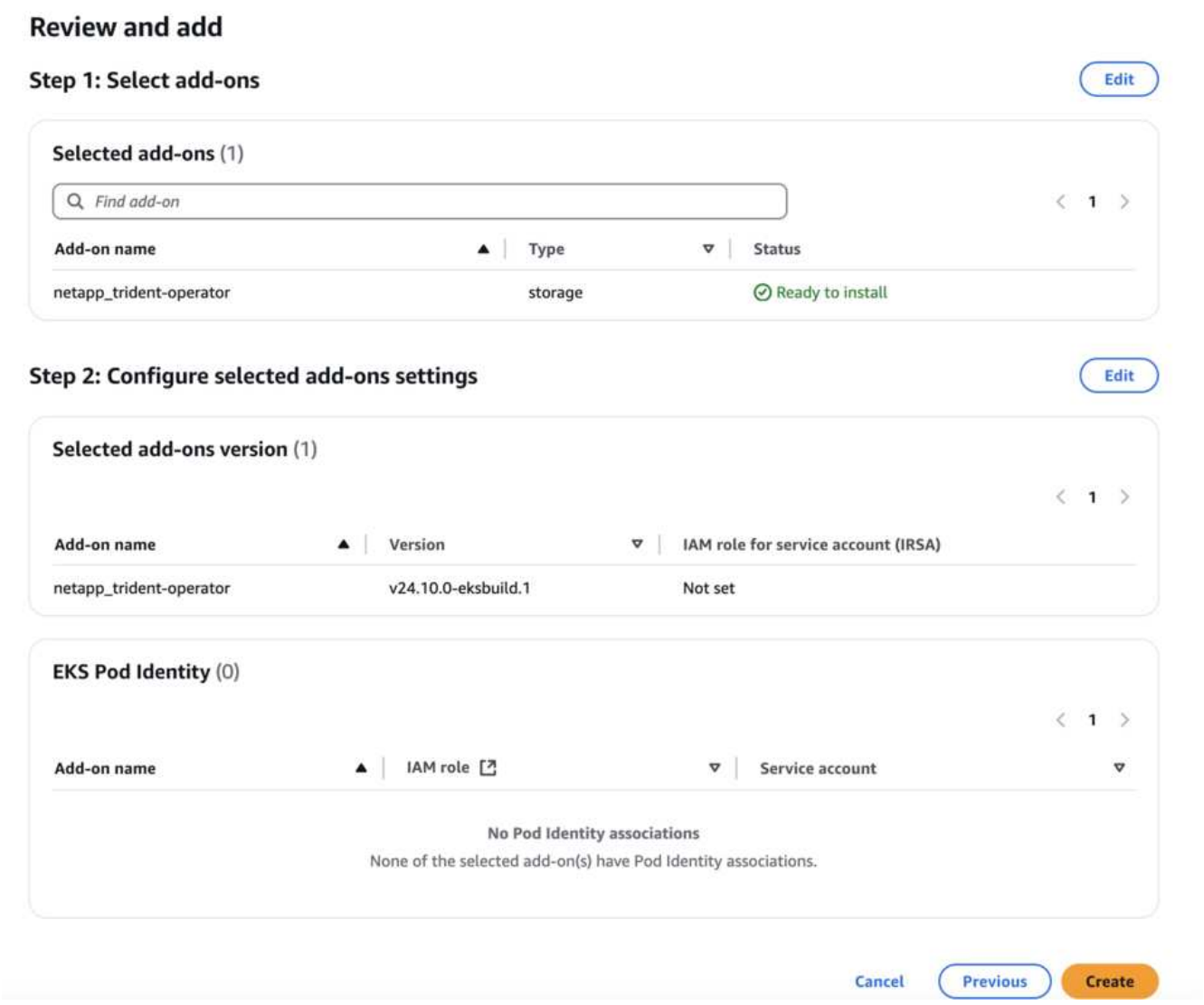
3. *AWS Marketplace 추가 기능*으로 이동하여 *storage* 카테고리를 선택합니다.



4. * NetApp Trident*를 찾아 Trident 추가 기능의 확인란을 선택하고 *다음*을 클릭합니다.
5. 원하는 애드온 버전을 선택하세요.

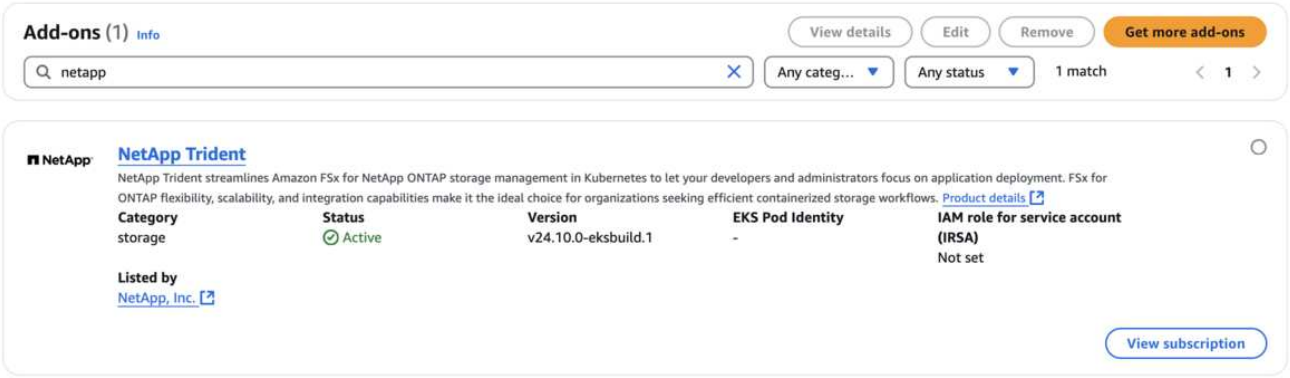


6. 필요한 추가 기능 설정을 구성합니다.



7. IRSA(서비스 계정에 대한 IAM 역할)를 사용하는 경우 추가 구성 단계를 참조하세요."여기" .
8. *만들기*를 선택하세요.

9. 추가 기능의 상태가 _활성_인지 확인하세요.



10. 다음 명령을 실행하여 Trident 클러스터에 제대로 설치되었는지 확인하세요.

```
kubectl get pods -n trident
```

11. 설정을 계속하고 스토리지 백엔드를 구성합니다. 자세한 내용은 다음을 참조하세요."스토리지 백엔드 구성" .

CLI를 사용하여 **Trident EKS** 애드온 설치/제거

CLI를 사용하여 **NetApp Trident EKS** 애드온을 설치하세요**.**

다음 예제 명령은 Trident EKS 추가 기능을 설치합니다.
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (전용 버전 포함)

CLI를 사용하여 **NetApp Trident EKS** 애드온을 제거합니다**.**

다음 명령은 Trident EKS 애드온을 제거합니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

kubectl로 백엔드 만들기

백엔드는 Trident 와 스토리지 시스템 간의 관계를 정의합니다. 이는 Trident 해당 스토리지 시스템과 통신하는 방법과 Trident 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Trident 설치한 후 다음 단계는 백엔드를 만드는 것입니다. 그만큼 `TridentBackendConfig` 사용자 정의 리소스 정의(CRD)를 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 만들고 관리할 수 있습니다. 다음을 사용하여 이 작업을 수행할 수 있습니다. `kubectl` 또는 Kubernetes 배포판에 맞는 동등한 CLI 도구입니다.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig` , `tbackendconfig`)는 Trident 백엔드를 관리할 수 있는 프런트엔드 네임스페이스 CRD입니다. `kubectl` . Kubernetes 및 스토리지 관리자는 이제 전용 명령줄 유틸리티가 필요 없이 Kubernetes CLI를 통해 직접 백엔드를 생성하고 관리할 수 있습니다.(`tridentctl`).

생성 시 `TridentBackendConfig` 객체의 경우 다음이 발생합니다.

- Trident 는 귀하가 제공한 구성을 기반으로 백엔드를 자동으로 생성합니다. 이는 내부적으로 다음과 같이 표현됩니다. `TridentBackend` (`tbe` , `tridentbackend`) CR.
- 그만큼 `TridentBackendConfig` 고유하게 결합됩니다 `TridentBackend` Trident 가 만든 것입니다.

각 `TridentBackendConfig` 일대일 매핑을 유지합니다. `TridentBackend` 전자는 사용자가 백엔드를 설계하고 구성하기 위해 제공하는 인터페이스이고, 후자는 Trident 실제 백엔드 객체를 표현하는 방식입니다.

> `TridentBackend`CR은 `Trident` 에 의해 자동으로 생성됩니다. 수정해서는 안 됩니다. 백엔드를 업데이트하려면 다음을 수정하세요. `TridentBackendConfig` 물체.

다음 예제를 참조하세요. `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

또한 다음 예제를 살펴볼 수도 있습니다. "트라이던트 설치 프로그램" 원하는 스토리지 플랫폼/서비스에 대한 샘플 구성을 위한 디렉토리입니다.

그만큼 `spec` 백엔드별 구성 매개변수를 사용합니다. 이 예에서 백엔드는 다음을 사용합니다. `ontap-san` 저장 드라이버이며 여기에 표로 정리된 구성 매개변수를 사용합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 다음을 참조하세요."스토리지 드라이버에 대한 백엔드 구성 정보" .

그만큼 `spec` 섹션에는 다음도 포함됩니다. `credentials` 그리고 `deletionPolicy` 새로 도입된 분야 `TridentBackendConfig` CR:

- `credentials`: 이 매개변수는 필수 필드이며 스토리지 시스템/서비스를 인증하는 데 사용되는 자격 증명을 포함합니다. 이는 사용자가 생성한 Kubernetes Secret으로 설정됩니다. 자격 증명은 일반 텍스트로 전달될 수 없으며 오류가 발생합니다.
- `deletionPolicy`: 이 필드는 다음과 같은 일이 발생해야 함을 정의합니다. `TridentBackendConfig` 삭제되었습니다. 다음 두 가지 값 중 하나를 취할 수 있습니다.
  - `delete`: 이로 인해 두 가지 모두 삭제됩니다. `TridentBackendConfig` CR 및 관련 백엔드. 이는 기본값입니다.
  - `retain`: 언제 `TridentBackendConfig` CR이 삭제되면 백엔드 정의는 여전히 존재하며 다음을 통해

관리할 수 있습니다. `tridentctl` . 삭제 정책을 다음으로 설정 `retain` 사용자가 이전 릴리스(21.04 이전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있도록 합니다. 이 필드의 값은 다음에 업데이트될 수 있습니다. `TridentBackendConfig` 생성됩니다.

백엔드의 이름은 다음을 사용하여 설정됩니다. `spec.backendName` . 지정하지 않으면 백엔드 이름이 다음 이름으로 설정됩니다. `TridentBackendConfig` 객체(메타데이터.이름). 백엔드 이름을 명시적으로 설정하는 것이 좋습니다. `spec.backendName` .

로 생성된 백엔드 `tridentctl` 연관된 것이 없습니다 `TridentBackendConfig` 물체. 이러한 백엔드를 관리하도록 선택할 수 있습니다. `kubectl` 생성하여 `TridentBackendConfig` 크.알. 동일한 구성 매개변수(예:)를 지정하는 데 주의해야 합니다. `spec.backendName` , `spec.storagePrefix` , `spec.storageDriverName` , 등). Trident 새로 생성된 항목을 자동으로 바인딩합니다. `TridentBackendConfig` 기존 백엔드를 사용합니다.

단계 개요

다음을 사용하여 새 백엔드를 생성하려면 `kubectl` , 다음을 수행해야 합니다.

1. 생성하다 "쿠버네티스 시크릿" . 비밀에는 Trident 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. 생성하다 `TridentBackendConfig` 물체. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 생성된 비밀을 참조합니다.

백엔드를 생성한 후에는 다음을 사용하여 상태를 관찰할 수 있습니다. `kubectl get tbc <tbc-name> -n <trident-namespace>` 그리고 추가적인 세부 정보를 수집합니다.

1단계: **Kubernetes Secret** 만들기

백엔드에 대한 액세스 자격 증명을 포함하는 비밀을 만듭니다. 이는 각 저장 서비스/플랫폼마다 고유합니다. 예를 들면 다음과 같습니다.

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

이 표는 각 저장 플랫폼의 비밀에 포함되어야 하는 필드를 요약한 것입니다.

저장 플랫폼 **Secret Fields** 설명	비밀	필드 설명
Azure NetApp Files	클라이언트ID	앱 등록의 클라이언트 ID
GCP용 Cloud Volumes Service	개인키_아이디	개인 키의 ID입니다. CVS 관리자 역할이 있는 GCP 서비스 계정의 API 키 일부
GCP용 Cloud Volumes Service	개인 키	개인 키. CVS 관리자 역할이 있는 GCP 서비스 계정의 API 키 일부
요소(NetApp HCI/ SolidFire)	엔드포인트	테넌트 자격 증명을 사용한 SolidFire 클러스터용 MVIP
ONTAP	사용자 이름	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됨
ONTAP	비밀번호	클러스터/SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됨
ONTAP	클라이언트 개인 키	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨
ONTAP	chapUsername	수신 사용자 이름. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 개시자 비밀. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapTargetUsername	대상 사용자 이름. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 대상 개시자 비밀. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`

이 단계에서 생성된 비밀은 다음에 참조됩니다. `spec.credentials` 의 분야 `TridentBackendConfig` 다음 단계에서 생성되는 객체입니다.

2단계: 만들기 `TridentBackendConfig` 씨.씨.

이제 다음을 생성할 준비가 되었습니다. `TridentBackendConfig` 크.알. 이 예에서는 다음을 사용하는 백엔드 `ontap-san` 드라이버는 다음을 사용하여 생성됩니다. `TridentBackendConfig` 아래에 표시된 객체:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

3단계: 상태 확인 `TridentBackendConfig` 씨.씨.

이제 당신이 생성했습니다 `TridentBackendConfig` CR, 상태를 확인할 수 있습니다. 다음 예를 참조하세요.

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                      BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

백엔드가 성공적으로 생성되어 바인딩되었습니다. `TridentBackendConfig` 크.알.

위상은 다음 값 중 하나를 취할 수 있습니다.

- `Bound`: 그 `TridentBackendConfig` CR은 백엔드와 연결되어 있으며 해당 백엔드에는 다음이 포함됩니다. `configRef` 로 설정 `TridentBackendConfig` CR의 uid.
- `Unbound`: 를 사용하여 표현됨 `""` . 그만큼 `TridentBackendConfig` 객체가 백엔드에 바인딩되지 않았습니다. 모두 새로 생성됨 `TridentBackendConfig` CR은 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 Unbound로 돌아갈 수 없습니다.
- `Deleting`: 그 `TridentBackendConfig` CR의 `deletionPolicy` 삭제되도록 설정되었습니다. 때 `TridentBackendConfig` CR이 삭제되면 삭제 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVC)이 없는 경우 삭제 `TridentBackendConfig` Trident 백엔드와 다음을

삭제하게 됩니다. `TridentBackendConfig` 크.알.

  - 백엔드에 하나 이상의 PVC가 있는 경우 삭제 상태로 전환됩니다. 그만큼 `TridentBackendConfig` CR도 이후 삭제 단계로 들어갑니다. 백엔드와 `TridentBackendConfig` 모든 PVC가 삭제된 후에만 삭제됩니다.

- `Lost`: 연관된 백엔드 `TridentBackendConfig` CR이 실수로 또는 의도적으로 삭제되었으며 `TridentBackendConfig` CR에는 삭제된 백엔드에 대한 참조가 여전히 있습니다. 그만큼 `TridentBackendConfig` CR은 여전히 삭제될 수 있습니다. `deletionPolicy` 값.
- `Unknown`: Trident 백엔드의 상태 또는 존재를 확인할 수 없습니다. `TridentBackendConfig` 크.알. 예를 들어, API 서버가 응답하지 않거나 `tridentbackends.trident.netapp.io` CRD가 없습니다. 여기에는 개입이 필요할 수 있습니다.

이 단계에서는 백엔드가 성공적으로 생성되었습니다! 추가로 처리할 수 있는 작업은 다음과 같습니다."백엔드 업데이트 및 백엔드 삭제" .

(선택 사항) 4단계: 자세한 내용 보기

다음 명령을 실행하면 백엔드에 대한 자세한 정보를 얻을 수 있습니다.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME         BACKEND UUID
PHASE    STATUS     STORAGE DRIVER    DELETION POLICY
backend-tbc-ontap-san    ontap-san-backend    8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success    ontap-san         delete
```

또한 YAML/JSON 덤프도 얻을 수 있습니다. `TridentBackendConfig` .

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo`포함하다 `backendName` 그리고 `backendUUID` 응답으로 생성된 백엔드의 `TridentBackendConfig` 크.알. 그만큼 `lastOperationStatus` 필드는 마지막 작업의 상태를 나타냅니다. `TridentBackendConfig` 사용자가 트리거할 수 있는 CR(예: 사용자가 무언가를 변경한 경우) `spec`) 또는 Trident에 의해 트리거됩니다(예: Trident 재시작 중). 성공이 될 수도 있고 실패가 될 수도 있습니다. `phase` 관계의 상태를 나타냅니다. `TridentBackendConfig` CR과 백엔드. 위의 예에서, `phase` Bound 값을 가지는데, 이는 다음을 의미합니다. `TridentBackendConfig` CR은 백엔드와 연관되어 있습니다.

당신은 실행할 수 있습니다 `kubectl -n trident describe tbc <tbc-cr-name>` 이벤트 로그의 세부 정보를 얻는 명령입니다.

> 연관된 백엔드를 포함하는 백엔드를 업데이트하거나 삭제할 수 없습니다. `TridentBackendConfig` 객체를 사용하여 `tridentctl` . 전환에 관련된 단계를 이해하려면 `tridentctl` 그리고 `TridentBackendConfig` ,"여기를 보세요" .

백엔드 관리

kubectl을 사용하여 백엔드 관리 수행

다음을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보세요. `kubectl` .

백엔드 삭제

삭제하여 `TridentBackendConfig` , Trident 백엔드를 삭제/보관하도록 지시합니다(기반 `deletionPolicy`). 백엔드를 삭제하려면 다음을 확인하세요. `deletionPolicy` 삭제되도록 설정되어 있습니다. 삭제하려면 `TridentBackendConfig` , 확인하십시오 `deletionPolicy` 유지되도록 설정되어 있습니다. 이렇게 하면 백엔드가 여전히 존재하고 다음을 사용하여 관리할 수 있습니다. `tridentctl` .

다음 명령을 실행하세요.

```
kubectl delete tbc <tbc-name> -n trident
```

Trident 사용 중이던 Kubernetes Secret을 삭제하지 않습니다. `TridentBackendConfig` . Kubernetes 사용자는 비밀을 정리할 책임이 있습니다. 비밀을 삭제할 때는 주의해야 합니다. 백엔드에서 사용하지 않는 비밀만 삭제해야 합니다.

기존 백엔드 보기

다음 명령을 실행하세요.

```
kubectl get tbc -n trident
```

또한 실행할 수도 있습니다 `tridentctl get backend -n trident` 또는 `tridentctl get backend -o yaml -n trident` 존재하는 모든 백엔드 목록을 얻습니다. 이 목록에는 또한 생성된 백엔드가 포함됩니다. `tridentctl` .

백엔드 업데이트

백엔드를 업데이트하는 데에는 여러 가지 이유가 있을 수 있습니다.

- 저장 시스템의 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 Kubernetes Secret이 사용됩니다. `TridentBackendConfig` 객체를 업데이트해야 합니다. Trident 제공된 최신 자격 증명으로 백엔드를 자동으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret을 업데이트합니다.

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- 매개변수(사용 중인 ONTAP SVM의 이름 등)를 업데이트해야 합니다.
  - 업데이트할 수 있습니다 `TridentBackendConfig` 다음 명령을 사용하여 Kubernetes를 통해 직접 객체를 가져올 수 있습니다.

    ```
    kubectl apply -f <updated-backend-file.yaml>
    ```

◦ 또는 기존 내용을 변경할 수 있습니다. `TridentBackendConfig` 다음 명령을 사용하여 CR을 실행합니다.

```
kubectl edit tbc <tbc-name> -n trident
```

- 백엔드 업데이트가 실패하면 백엔드는 마지막으로 알려진 구성을 그대로 유지합니다. 원인을 확인하려면 다음을 실행하여 로그를 볼 수 있습니다. `kubectl get tbc <tbc-name> -o yaml -n trident` 또는 `kubectl describe tbc <tbc-name> -n trident` .
- 구성 파일의 문제를 파악하고 수정한 후 업데이트 명령을 다시 실행할 수 있습니다.

tridentctl로 백엔드 관리 수행

다음을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보세요. `tridentctl` .

백엔드 만들기

생성한 후"백엔드 구성 파일" 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file> -n trident
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후에는 간단히 다음을 실행할 수 있습니다. `create` 다시 명령을 내리세요.

백엔드 삭제

Trident 에서 백엔드를 삭제하려면 다음을 수행하세요.

1. 백엔드 이름을 검색합니다.

```
tridentctl get backend -n trident
```

2. 백엔드를 삭제합니다.

```
tridentctl delete backend <backend-name> -n trident
```

Trident 이 백엔드에서 여전히 존재하는 볼륨과 스냅샷을 프로비저닝한 경우 백엔드를 삭제하면 새 볼륨을 프로비저닝할 수 없습니다. 백엔드는 "삭제 중" 상태로 계속 존재합니다.

기존 백엔드 보기

Trident 알고 있는 백엔드를 보려면 다음을 수행하세요.

- 요약을 보려면 다음 명령을 실행하세요.

```
tridentctl get backend -n trident
```

- 모든 세부 정보를 보려면 다음 명령을 실행하세요.

```
tridentctl get backend -o json -n trident
```

백엔드 업데이트

새로운 백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

백엔드 업데이트가 실패한 경우 백엔드 구성에 문제가 있거나 잘못된 업데이트를 시도한 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후에는 간단히 다음을 실행할 수 있습니다. `update` 다시 명령을 내리세요.

백엔드를 사용하는 스토리지 클래스를 식별합니다.

이는 JSON으로 답할 수 있는 질문 종류의 예입니다. `tridentctl` 백엔드 객체에 대한 출력. 이것은 다음을 사용합니다 `jq` 설치해야 하는 유틸리티입니다.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

이는 다음을 사용하여 생성된 백엔드에도 적용됩니다. `TridentBackendConfig` .

백엔드 관리 옵션 간 이동

Trident 에서 백엔드를 관리하는 다양한 방법에 대해 알아보세요.

백엔드 관리를 위한 옵션

의 도입으로 `TridentBackendConfig` 이제 관리자는 백엔드를 관리하는 두 가지 고유한 방법을 사용할 수 있습니다. 이는 다음과 같은 질문을 제기합니다.

- 백엔드를 사용하여 생성할 수 있습니까? `tridentctl` ~로 관리되다 `TridentBackendConfig` ?
- 백엔드를 사용하여 생성할 수 있습니까? `TridentBackendConfig` 사용하여 관리됩니다 `tridentctl` ?

관리하다 `tridentctl` 백엔드를 사용하여 `TridentBackendConfig`

이 섹션에서는 다음을 사용하여 생성된 백엔드를 관리하는 데 필요한 단계를 다룹니다. `tridentctl` Kubernetes 인터페이스를 통해 직접 생성 `TridentBackendConfig` 사물.

이는 다음 시나리오에 적용됩니다.

- 기존 백엔드가 없는 경우 `TridentBackendConfig` 왜냐하면 그들은 ~로 창조되었기 때문이다 `tridentctl` .
- 새로운 백엔드가 생성되었습니다. `tridentctl` , 다른 `TridentBackendConfig` 객체가 존재합니다.

두 시나리오 모두 백엔드는 계속 존재하며 Trident 볼륨을 예약하고 이를 운영합니다. 관리자는 다음 두 가지 선택 중 하나를 선택할 수 있습니다.

- 계속 사용 `tridentctl` 이를 사용하여 생성된 백엔드를 관리합니다.
- 다음을 사용하여 생성된 백엔드 바인딩 `tridentctl` 새로운 것에 `TridentBackendConfig` 물체. 그렇게 하면 백엔드가 다음을 사용하여 관리됩니다. `kubectl` 그리고 아니다 `tridentctl` .

기존 백엔드를 관리하려면 다음을 사용합니다. `kubectl` , 당신은 만들어야 할 것입니다 `TridentBackendConfig` 기존 백엔드에 바인딩됩니다. 작동 원리는 다음과 같습니다.

1. Kubernetes Secret을 생성합니다. 비밀에는 Trident 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. 생성하다 `TridentBackendConfig` 물체. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 생성된 비밀을 참조합니다. 동일한 구성 매개변수(예:)를 지정하는 데 주의해야 합니다. `spec.backendName` , `spec.storagePrefix` , `spec.storageDriverName` , 등). `spec.backendName` 기존 백엔드의 이름으로 설정해야 합니다.

0단계**:** 백엔드 식별

생성하려면 `TridentBackendConfig` 기존 백엔드에 바인딩하려면 백엔드 구성을 얻어야 합니다. 이 예에서는 다음 JSON 정의를 사용하여 백엔드가 생성되었다고 가정해 보겠습니다.

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                  UUID
| STATE   | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

1단계: **Kubernetes Secret** 만들기

다음 예와 같이 백엔드에 대한 자격 증명을 포함하는 비밀을 만듭니다.

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

2단계: 만들기 `TridentBackendConfig` 씨.씨.

다음 단계는 다음을 만드는 것입니다. `TridentBackendConfig` 기존에 자동으로 바인딩되는 CR `ontap-nas-backend` (이 예에서처럼). 다음 요구 사항이 충족되는지 확인하세요.

- 동일한 백엔드 이름이 정의되어 있습니다. `spec.backendName` .
- 구성 매개변수는 원래 백엔드와 동일합니다.
- 가상 풀(존재하는 경우)은 원래 백엔드와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우에는 `TridentBackendConfig` 다음과 같이 보일 것입니다:

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

3단계: 상태 확인 `TridentBackendConfig` 씨.씨.

후에 `TridentBackendConfig` 생성되었으므로 해당 단계는 다음과 같아야 합니다. `Bound` . 또한 기존 백엔드와 동일한 백엔드 이름과 UUID를 반영해야 합니다.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

백엔드는 이제 다음을 사용하여 완전히 관리됩니다. `tbc-ontap-nas-backend` `TridentBackendConfig` 물체.

관리하다 `TridentBackendConfig` 백엔드를 사용하여 `tridentctl`

`tridentctl`를 사용하여 생성된 백엔드를 나열하는 데 사용할 수 있습니다. `TridentBackendConfig` . 또한 관리자는 다음을 통해 이러한 백엔드를 완전히 관리하도록 선택할 수도 있습니다. `tridentctl` 삭제하여 `TridentBackendConfig` 그리고 확인하다 `spec.deletionPolicy` 로 설정됩니다 `retain` .

0단계: 백엔드 식별

예를 들어, 다음 백엔드가 다음을 사용하여 생성되었다고 가정해 보겠습니다. `TridentBackendConfig` :

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

출력에서 다음이 표시됩니다. `TridentBackendConfig` 성공적으로 생성되었으며 백엔드에 바인딩되었습니다. [백엔드의 UUID를 확인하세요]

1단계: 확인 `deletionPolicy` 로 설정됩니다 `retain`

의 가치를 살펴보자 `deletionPolicy` . 이것은 설정되어야 합니다 `retain` . 이것은 다음을 보장합니다. `TridentBackendConfig` CR이 삭제되면 백엔드 정의는 여전히 존재하며 다음을 통해 관리할 수 있습니다. `tridentctl` .

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

다음 단계로 진행하지 마십시오. `deletionPolicy` 로 설정됩니다 `retain` .

2단계: 삭제 `TridentBackendConfig` 씨.씨.

마지막 단계는 삭제하는 것입니다. `TridentBackendConfig` 크.알. 확인 후 `deletionPolicy` 로 설정됩니다 `retain` , 삭제를 진행할 수 있습니다.

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

삭제 시 `TridentBackendConfig` 객체를 제거하면 Trident 백엔드 자체를 삭제하지 않고 객체를 제거합니다.

스토리지 클래스 생성 및 관리

스토리지 클래스 생성

Kubernetes StorageClass 객체를 구성하고 Trident 가 볼륨을 프로비저닝하는 방법을 지시하는 스토리지 클래스를 만듭니다.

Kubernetes StorageClass 객체 구성

그만큼 "Kubernetes StorageClass 객체" Trident 해당 클래스에 사용되는 프로비저너로 식별하고 Trident 볼륨을 프로비저닝하는 방법을 지시합니다. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

스토리지 클래스 생성

StorageClass 객체를 만든 후에는 스토리지 클래스를 만들 수 있습니다. 저장 클래스 샘플 사용하거나 수정할 수 있는 몇 가지 기본 샘플을 제공합니다.

단계

1. 이것은 Kubernetes 객체이므로 다음을 사용합니다. `kubectl` Kubernetes에서 생성하세요.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스를 볼 수 있어야 하며, Trident 가 백엔드에서 풀을 검색했어야 합니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```


저장 클래스 샘플

Trident 제공합니다 "특정 백엔드에 대한 간단한 스토리지 클래스 정의" .

또는 편집할 수 있습니다 `sample-input/storage-class-csi.yaml.templ` 설치 프로그램과 함께 제공되는 파일을 교체합니다. *`BACKEND_TYPE`* 저장소 드라이버 이름으로.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

스토리지 클래스 관리

기존 스토리지 클래스를 보고, 기본 스토리지 클래스를 설정하고, 스토리지 클래스 백엔드를 식별하고, 스토리지 클래스를 삭제할 수 있습니다.

기존 스토리지 클래스 보기

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass <storage-class> -o json
```

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass <storage-class> -o json
```

기본 저장소 클래스 설정

Kubernetes 1.6에서는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 이는 사용자가 PVC(영구 볼륨 클레임)에서 영구 볼륨을 지정하지 않은 경우 영구 볼륨을 프로비저닝하는 데 사용되는 스토리지 클래스입니다.

- 주석을 설정하여 기본 저장 클래스를 정의합니다. `storageclass.kubernetes.io/is-default-class` 저장 클래스 정의에서 true로 설정합니다. 사양에 따르면, 다른 값이나 주석이 없는 경우 거짓으로 해석됩니다.
- 다음 명령을 사용하여 기존 스토리지 클래스를 기본 스토리지 클래스로 구성할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 마찬가지로 다음 명령을 사용하여 기본 저장소 클래스 주석을 제거할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 설치 프로그램 번들에는 이 주석이 포함된 예도 있습니다.

> 클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes는 기술적으로 두 개 이상을 갖는 것을 막지는 않지만, 기본 스토리지 클래스가 전혀 없는 것처럼 동작합니다.

스토리지 클래스의 백엔드 식별

이는 JSON으로 답할 수 있는 질문 종류의 예입니다. `tridentctl` Trident 백엔드 객체에 대한 출력. 이것은 다음을 사용합니다 `jq` 먼저 설치해야 할 유틸리티가 있습니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

스토리지 클래스 삭제

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행하세요.

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`저장 클래스로 대체해야 합니다.

이 스토리지 클래스를 통해 생성된 모든 영구 볼륨은 변경되지 않으며 Trident 계속해서 이를 관리합니다.

> Trident 공백을 강화합니다. `fsType` 그것이 만들어내는 볼륨 때문에요. iSCSI 백엔드의 경우 다음을 적용하는 것이 좋습니다. `parameters.fsType` StorageClass에서. 기존 StorageClass를 삭제하고 다시 생성해야 합니다. `parameters.fsType` 지정된.

볼륨 제공 및 관리

볼륨 제공

구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim(PVC)을 만듭니다. 그런 다음 PV를 포드에 장착할 수 있습니다.

개요

에이 "지속적 볼륨 클레임" (PVC)는 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

PVC는 특정 크기 또는 액세스 모드의 저장소를 요청하도록 구성될 수 있습니다. 연관된 StorageClass를 사용하면 클러스터 관리자는 PersistentVolume 크기 및 액세스 모드(성능이나 서비스 수준 등) 이상을 제어할 수 있습니다.

PVC를 만든 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 만듭니다

단계

1. PVC을 생성합니다.

```
kubectl create -f pvc.yaml
```

2. PVC 상태를 확인하세요.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage Bound    pv-name    1Gi        RWO                    5m
```

1. 볼륨을 포드에 마운트합니다.

```
kubectl create -f pv-pod.yaml
```

> 다음을 사용하여 진행 상황을 모니터링할 수 있습니다. `kubectl get pod --watch` .

2. 볼륨이 마운트되었는지 확인하세요. `/my/mount/path` .

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. 이제 Pod를 삭제할 수 있습니다. Pod 애플리케이션은 더 이상 존재하지 않지만 볼륨은 그대로 유지됩니다.

```
kubectl delete pod pv-pod
```

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

다음 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWO 접근이 가능한 **PVC**

이 예에서는 StorageClass와 연관된 RWO 액세스가 있는 기본 PVC를 보여줍니다. `basic-csi` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

NVMe/TCP를 사용한 **PVC**

이 예에서는 RWO 액세스가 있는 NVMe/TCP용 기본 PVC를 보여줍니다. 이 PVC는 StorageClass와 연결되어 있습니다. `protection-gold` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Pod 매니페스트 샘플

이러한 예는 PVC를 포드에 부착하는 기본 구성을 보여줍니다.

기본 구성

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

기본 **NVMe/TCP** 구성

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

볼륨 확장

Trident Kubernetes 사용자에게 볼륨을 생성한 후에 볼륨을 확장할 수 있는 기능을 제공합니다. iSCSI, NFS, SMB, NVMe/TCP 및 FC 볼륨을 확장하는 데 필요한 구성에 대한 정보를 찾아보세요.

iSCSI 볼륨 확장

CSI 프로비저너를 사용하여 iSCSI 영구 볼륨(PV)을 확장할 수 있습니다.

> iSCSI 볼륨 확장은 다음에서 지원됩니다. `ontap-san` , `ontap-san-economy` , `solidfire-san` 드라이버가 필요하며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

StorageClass 정의를 편집하여 다음을 설정합니다. `allowVolumeExpansion` 필드로 `true` .

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 다음을 포함하도록 편집합니다. `allowVolumeExpansion` 매개변수.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다.

PVC 정의를 편집하고 업데이트합니다. `spec.resources.requests.storage` 새로 원하는 크기를 반영해야 하며, 원래 크기보다 커야 합니다.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident 영구 볼륨(PV)을 생성하고 이를 영구 볼륨 클레임(PVC)과 연결합니다.

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

3단계: **PVC**를 부착하는 포드 정의

PV를 포드에 부착하여 크기를 조절할 수 있습니다. iSCSI PV 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결되어 있으면 Trident 스토리지 백엔드의 볼륨을 확장하고, 장치를 다시 스캔하고, 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident 스토리지 백엔드의 볼륨을 확장합니다. PVC가 포드에 바인딩되면 Trident 장치를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 다음을 사용하는 포드가 생성됩니다. `san-pvc` .

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

4단계: **PV** 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 업데이트하십시오. `spec.resources.requests.storage` 2Gi로.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

5단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 작동하는지 확인할 수 있습니다.

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

FC 볼륨 확장

CSI 프로비저너를 사용하여 FC 영구 볼륨(PV)을 확장할 수 있습니다.

FC 볼륨 확장은 다음에 의해 지원됩니다. `ontap-san` 드라이버이며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

StorageClass 정의를 편집하여 다음을 설정합니다. `allowVolumeExpansion` 필드로 `true` .

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 다음을 포함하도록 편집합니다. `allowVolumeExpansion` 매개변수.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다.

PVC 정의를 편집하고 업데이트합니다. `spec.resources.requests.storage` 새로 원하는 크기를 반영해야 하며, 원래 크기보다 커야 합니다.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident 영구 볼륨(PV)을 생성하고 이를 영구 볼륨 클레임(PVC)과 연결합니다.

```
kubectl get pvc
NAME       STATUS    VOLUME                                     CAPACITY
ACCESS MODES    STORAGECLASS    AGE
san-pvc    Bound     pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO             ontap-san       8s

kubectl get pv
NAME                                       CAPACITY    ACCESS MODES
RECLAIM POLICY    STATUS    CLAIM              STORAGECLASS    REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi         RWO
Delete            Bound     default/san-pvc    ontap-san                 10s
```

3단계: **PVC**를 부착하는 포드 정의

PV를 포드에 부착하여 크기를 조절할 수 있습니다. FC PV 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결되어 있으면 Trident 스토리지 백엔드의 볼륨을 확장하고, 장치를 다시 스캔하고, 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident 스토리지 백엔드의 볼륨을 확장합니다. PVC가 포드에 바인딩되면 Trident 장치를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 다음을 사용하는 포드가 생성됩니다. `san-pvc` .

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

4단계: **PV** 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 업데이트하십시오. `spec.resources.requests.storage` 2Gi로.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

5단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 작동하는지 확인할 수 있습니다.

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

NFS 볼륨 확장

Trident NFS PV에 대한 볼륨 확장을 지원합니다. `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `gcp-cvs` , 그리고 `azure-netapp-files` 백엔드.

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

NFS PV 크기를 조정하려면 관리자는 먼저 볼륨 확장을 허용하도록 스토리지 클래스를 구성해야 합니다. `allowVolumeExpansion` 필드로 `true` :

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

이 옵션 없이 이미 스토리지 클래스를 생성한 경우 다음을 사용하여 기존 스토리지 클래스를 간단히 편집할 수 있습니다. `kubectl edit storageclass` 볼륨 확장을 허용합니다.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다.

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES    STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

3단계: **PV** 확장

새로 생성된 20 MiB PV를 1 GiB로 크기를 조정하려면 PVC를 편집하고 설정하세요. `spec.resources.requests.storage` 1GiB까지:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

4단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 크기 조정이 올바르게 수행되었는지 확인할 수 있습니다.

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

수입량

기존 스토리지 볼륨을 Kubernetes PV로 가져올 수 있습니다. `tridentctl import`.

개요 및 고려 사항

볼륨을 Trident 로 가져와서 다음을 수행할 수 있습니다.

- 애플리케이션을 컨테이너화하고 기존 데이터 세트를 재사용합니다.
- 임시 애플리케이션에 데이터 세트 복제본 사용
- 실패한 Kubernetes 클러스터 재구축
- 재해 복구 중 애플리케이션 데이터 마이그레이션

고려 사항

볼륨을 가져오기 전에 다음 고려 사항을 검토하세요.

- Trident RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 대상 볼륨입니다. 볼륨을 Trident 로 가져오기 전에 미러 관계를 해제해야 합니다.

- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 현재 사용되는 볼륨을 가져오려면 볼륨을 복제한 다음 가져오기를 수행합니다.

  이는 블록 볼륨의 경우 특히 중요합니다. Kubernetes는 이전 연결을 인식하지 못하고 활성 볼륨을 Pod에 쉽게 연결할 수 있기 때문입니다. 이로 인해 데이터가 손상될 수 있습니다.

- 그렇지만 `StorageClass` PVC에서는 이 매개변수를 지정해야 하지만, Trident 가져오기 중에 이 매개변수를 사용하지 않습니다. 스토리지 클래스는 볼륨을 생성하는 동안 스토리지 특성에 따라 사용 가능한 풀을 선택하는 데 사용됩니다. 볼륨이 이미 존재하므로 가져오는 동안 풀을 선택할 필요가 없습니다. 따라서 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드나 풀에 볼륨이 존재하더라도 가져오기가 실패하지 않습니다.
- 기존 볼륨 크기가 결정되어 PVC에 설정됩니다. 볼륨이 스토리지 드라이버에 의해 가져온 후, PV는 PVC에 대한 ClaimRef와 함께 생성됩니다.
  - 회수 정책은 처음에 다음과 같이 설정됩니다. `retain` PV에서. Kubernetes가 PVC와 PV를 성공적으로 바인딩한 후, 회수 정책이 스토리지 클래스의 회수 정책과 일치하도록 업데이트됩니다.
  - 스토리지 클래스의 회수 정책이 있는 경우 `delete` PV가 삭제되면 저장 볼륨도 삭제됩니다.
- 기본적으로 Trident PVC를 관리하고 백엔드에서 FlexVol volume 과 LUN의 이름을 변경합니다. 당신은 통과 할 수 있습니다 `--no-manage` 관리되지 않는 볼륨을 가져오기 위한 플래그입니다. 당신이 사용하는 경우 `--no -manage` Trident 객체의 수명 주기 동안 PVC 또는 PV에 대한 추가 작업을 수행하지 않습니다. PV가 삭제되어도 저장 볼륨은 삭제되지 않으며 볼륨 복제 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다.

  이 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하지만 그렇지 않은 경우 Kubernetes 외부에서 스토리지 볼륨의 수명 주기를 관리하려는 경우에 유용합니다.

- PVC와 PV에 주석이 추가되어 볼륨이 가져왔는지, PVC와 PV가 관리되는지 여부를 나타내는 두 가지 목적을 갖습니다. 이 주석은 수정하거나 제거해서는 안 됩니다.

볼륨 가져오기

사용할 수 있습니다 `tridentctl import` 볼륨을 가져오려면.

단계

1. 영구 볼륨 클레임(PVC) 파일을 만듭니다(예: `pvc.yaml`) PVC를 만드는 데 사용됩니다. PVC 파일에는 다음이 포함되어야 합니다. `name` , `namespace` , `accessModes` , 그리고 `storageClassName` . 선택적으로 지정할 수 있습니다. `unixPermissions` PVC 정의에서.

   다음은 최소 사양의 예입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

PV 이름이나 볼륨 크기와 같은 추가 매개변수를 포함하지 마세요. 이로 인해 가져오기 명령이 실패할 수 있습니다.

2. 사용하다 `tridentctl import` 볼륨을 포함하는 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume, Cloud Volumes Service 경로)을 지정하는 명령입니다. 그만큼 `-f` PVC 파일의 경로를 지정하려면 인수가 필요합니다.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

예시

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하세요.

ONTAP NAS 및 ONTAP NAS FlexGroup

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `ontap-nas` 그리고 `ontap-nas-flexgroup` 운전자.

- Trident 다음을 사용하여 볼륨 가져오기를 지원하지 않습니다. `ontap-nas-economy` 운전사.
- 그만큼 `ontap-nas` 그리고 `ontap-nas-flexgroup` 드라이버는 중복된 볼륨 이름을 허용하지 않습니다.

각 볼륨은 다음과 같이 생성됩니다. `ontap-nas` 드라이버는 ONTAP 클러스터의 FlexVol volume 입니다. FlexVol 볼륨을 가져오는 방법 `ontap-nas` 드라이버도 동일하게 작동합니다. ONTAP 클러스터에 이미 존재하는 FlexVol 볼륨은 다음과 같이 가져올 수 있습니다. `ontap-nas` PVC. 마찬가지로 FlexGroup 볼륨은 다음과 같이 가져올 수 있습니다. `ontap-nas-flexgroup` PVC.

ONTAP NAS 예시

다음은 관리되는 볼륨과 관리되지 않는 볼륨 가져오기의 예를 보여줍니다.

관리되는 볼륨

다음 예제에서는 이름이 지정된 볼륨을 가져옵니다. `managed_volume` 백엔드에 이름이 지정됨 `ontap_nas` :

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

사용 시 `--no-manage` 인수에 따라 Trident 볼륨의 이름을 바꾸지 않습니다.

다음 예제에서는 다음을 가져옵니다. `unmanaged_volume` 에 `ontap_nas` 백엔드:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

ONTAP 산

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `ontap-san` (iSCSI, NVMe/TCP 및 FC) 및 `ontap-san-economy` 운전자.

Trident 단일 LUN을 포함하는 ONTAP SAN FlexVol 볼륨을 가져올 수 있습니다. 이는 다음과 일치합니다. `ontap-san` 각 PVC에 대한 FlexVol volume 과 FlexVol volume 내의 LUN을 생성하는 드라이버입니다. Trident FlexVol volume 가져와 PVC 정의와 연결합니다. Trident 수입이 가능합니다 `ontap-san-economy` 여러 LUN을 포함하는

볼륨.

ONTAP SAN 예시

다음은 관리되는 볼륨과 관리되지 않는 볼륨 가져오기의 예를 보여줍니다.

관리되는 볼륨

관리되는 볼륨의 경우 Trident FlexVol volume 이름을 다음과 같이 변경합니다. `pvc-<uuid>` FlexVol volume 내의 LUN과 포맷 `lun0` .

다음 예제에서는 다음을 가져옵니다. `ontap-san-managed` FlexVol volume 이 존재합니다. `ontap_san_default` 백엔드:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

다음 예제에서는 다음을 가져옵니다. `unmanaged_example_volume` 에 `ontap_san` 백엔드:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

다음 예와 같이 Kubernetes 노드 IQN과 IQN을 공유하는 igroup에 LUN이 매핑된 경우 오류가 발생합니다. `LUN`

`already mapped to initiator(s) in this group`. 볼륨을 가져오려면 초기자를 제거하거나 LUN의 매핑을 해제해야 합니다.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

요소

Trident 다음을 사용하여 NetApp Element 소프트웨어 및 NetApp HCI 볼륨 가져오기를 지원합니다. `solidfire-san` 운전사.

> Element 드라이버는 중복된 볼륨 이름을 지원합니다. 그러나 볼륨 이름이 중복되면 Trident 오류를 반환합니다. 해결 방법으로 볼륨을 복제하고 고유한 볼륨 이름을 제공한 다음 복제된 볼륨을 가져옵니다.

요소 예

다음 예제에서는 다음을 가져옵니다. `element-managed` 백엔드의 볼륨 `element_default` .

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

구글 클라우드 플랫폼

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `gcp-cvs` 운전사.

> Google Cloud Platform에서 NetApp Cloud Volumes Service 지원하는 볼륨을 가져오려면 볼륨 경로로 볼륨을 식별합니다. 볼륨 경로는 볼륨 내보내기 경로의 일부입니다. `:/` . 예를 들어, 내보내기 경로가 `10.0.0.1:/adroit-jolly-swift` , 볼륨 경로는 `adroit-jolly-swift` .

Google Cloud Platform 예시

다음 예제에서는 다음을 가져옵니다. `gcp-cvs` 백엔드의 볼륨 `gcpcvs_YEppr` 볼륨 경로와 함께 `adroit-jolly-swift` .

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `azure-netapp-files` 운전사.

> Azure NetApp Files 볼륨을 가져오려면 볼륨 경로로 볼륨을 식별합니다. 볼륨 경로는 볼륨 내보내기 경로의 일부입니다. `:/` . 예를 들어, 마운트 경로가 `10.0.0.2:/importvol1` , 볼륨 경로는 `importvol1` .

Azure NetApp Files 예제

다음 예제에서는 다음을 가져옵니다. `azure-netapp-files` 백엔드의 볼륨 `azurenetappfiles_40517` 볼륨 경로로 `importvol1` .

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `google-cloud-netapp-volumes` 운전사.

Google Cloud NetApp Volumes 예시

다음 예제에서는 다음을 가져옵니다. `google-cloud-netapp-volumes` 백엔드의 볼륨 `backend-tbc-gcnv1` 볼륨과 함께 `testvoleasiaeast1` .

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

다음 예제에서는 다음을 가져옵니다. `google-cloud-netapp-volumes` 동일한 영역에 두 개의 볼륨이 존재할 때의 볼륨:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

볼륨 이름 및 레이블 사용자 정의

Trident 사용하면 생성한 볼륨에 의미 있는 이름과 레이블을 지정할 수 있습니다. 이를 통해 볼륨을 식별하고 해당 Kubernetes 리소스(PVC)에 쉽게 매핑할 수 있습니다. 사용자 정의 볼륨 이름과 사용자 정의 레이블을 생성하기 위해 백엔드 수준에서 템플릿을 정의할 수도 있습니다. 생성, 가져오기 또는 복제하는 모든 볼륨은 템플릿을 준수합니다.

시작하기 전에

사용자 정의 가능한 볼륨 이름 및 레이블 지원:

1. 볼륨 생성, 가져오기 및 복제 작업.
2. ontap-nas-economy 드라이버의 경우, Qtree 볼륨의 이름만 이름 템플릿을 따릅니다.
3. ontap-san-economy 드라이버의 경우 LUN 이름만 이름 템플릿을 따릅니다.

제한 사항

1. 사용자 정의 가능한 볼륨 이름은 ONTAP 온프레미스 드라이버와만 호환됩니다.
2. 사용자 정의 가능한 볼륨 이름은 기존 볼륨에는 적용되지 않습니다.

사용자 정의 가능한 볼륨 이름의 주요 동작

1. 이름 템플릿의 구문이 잘못되어 오류가 발생하면 백엔드 생성이 실패합니다. 그러나 템플릿 애플리케이션이 실패하면 볼륨 이름은 기존 명명 규칙에 따라 지정됩니다.
2. 백엔드 구성의 이름 템플릿을 사용하여 볼륨의 이름을 지정한 경우 스토리지 접두사는 적용되지 않습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블을 사용한 백엔드 구성 예

사용자 정의 이름 템플릿은 루트 및/또는 풀 수준에서 정의할 수 있습니다.

루트 수준 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

풀 레벨 예시

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

이름 템플릿 예시

예시 **1**:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

예시 **2**:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

고려할 사항

1. 볼륨 가져오기의 경우, 기존 볼륨에 특정 형식의 레이블이 있는 경우에만 레이블이 업데이트됩니다. 예를 들어: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. 관리형 볼륨 가져오기의 경우 볼륨 이름은 백엔드 정의의 루트 수준에서 정의된 이름 템플릿을 따릅니다.
3. Trident 저장 접두사와 함께 슬라이스 연산자를 사용하는 것을 지원하지 않습니다.
4. 템플릿에서 고유한 볼륨 이름이 생성되지 않으면 Trident 몇 개의 무작위 문자를 추가하여 고유한 볼륨 이름을 생성합니다.
5. NAS Economy 볼륨의 사용자 지정 이름이 64자를 초과하는 경우 Trident 기존 명명 규칙에 따라 볼륨 이름을 지정합니다. 다른 모든 ONTAP 드라이버의 경우 볼륨 이름이 이름 제한을 초과하면 볼륨 생성 프로세스가 실패합니다.

네임스페이스 간에 **NFS** 볼륨 공유

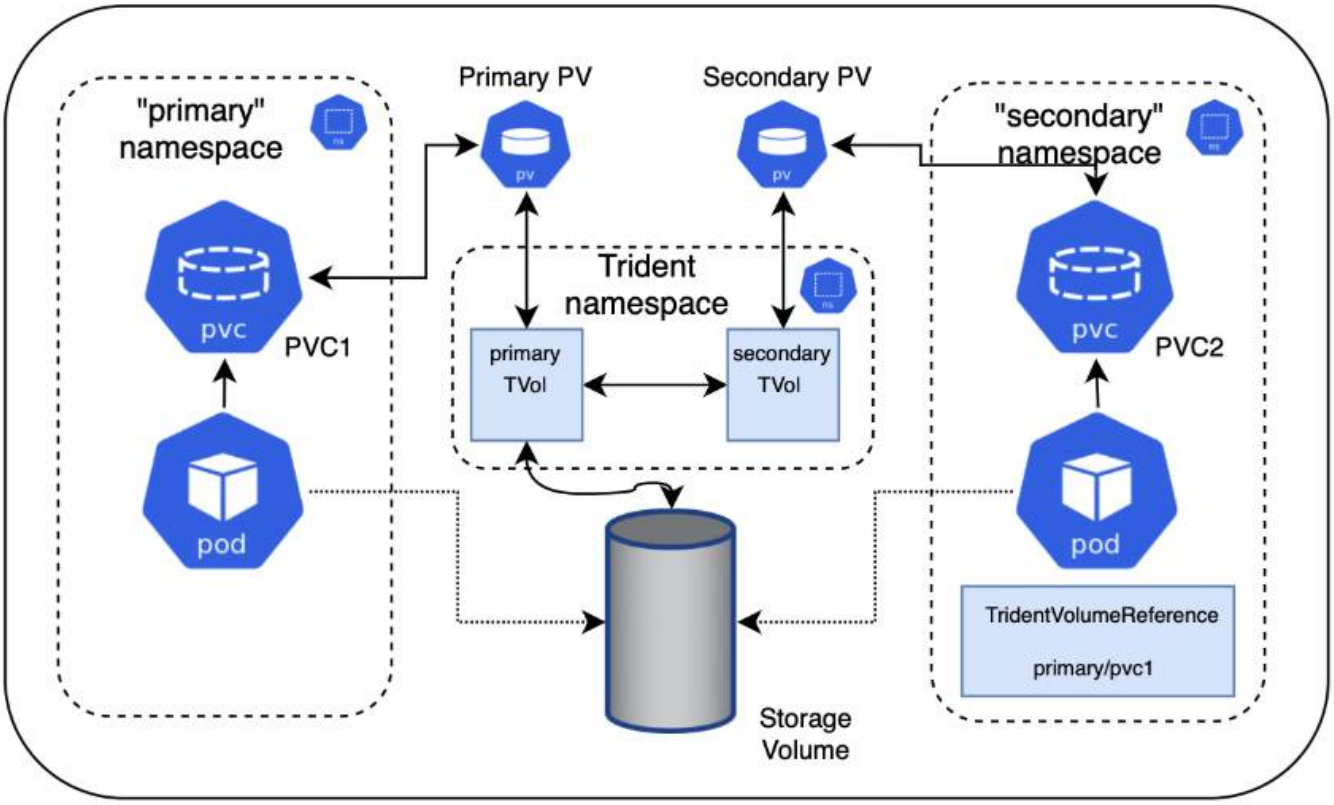
Trident 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

특징

TridentVolumeReference CR을 사용하면 하나 이상의 Kubernetes 네임스페이스에서 ReadWriteMany(RWX) NFS 볼륨을 안전하게 공유할 수 있습니다. 이 Kubernetes 기반 솔루션은 다음과 같은 이점을 제공합니다.

- 보안을 보장하기 위한 다양한 수준의 액세스 제어
- 모든 Trident NFS 볼륨 드라이버와 함께 작동합니다.
- tridentctl이나 기타 비네이티브 Kubernetes 기능에 대한 의존성 없음

이 다이어그램은 두 개의 Kubernetes 네임스페이스에서 NFS 볼륨을 공유하는 것을 보여줍니다.



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1 볼륨을 공유하도록 소스 **PVC**를 구성합니다**.**

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.

2 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여합니다**.**

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다**.**

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4 대상 네임스페이스에 하위 **PVC**를 만듭니다**.**

대상 네임스페이스의 소유자는 소스 PVC의 데이터 소스를 사용하기 위해 하위 PVC를 생성합니다.

소스 및 대상 네임스페이스 구성

보안을 보장하려면 네임스페이스 간 공유에 소스 네임스페이스 소유자, 클러스터 관리자, 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계마다 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: PVC를 생성합니다.(`pvc1`) 대상 네임스페이스와 공유할 수 있는 권한을 부여하는 소스 네임스페이스에서(`namespace2`)를 사용하여 `shareToNamespace` 주석.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident PV와 백엔드 NFS 스토리지 볼륨을 생성합니다.

- 쉼표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. 예를 들어, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` .
- 다음을 사용하여 모든 네임스페이스에 공유할 수 있습니다. `*` . 예를 들어, `trident.netapp.io/shareToNamespace: *`
- PVC를 업데이트하여 다음을 포함할 수 있습니다. `shareToNamespace` 언제든지 주석을 달 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하세요.
3. 대상 네임스페이스 소유자: 소스 네임스페이스를 참조하는 대상 네임스페이스에 TridentVolumeReference CR을 생성합니다. `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 대상 네임스페이스 소유자: PVC 생성(`pvc2`) 대상 네임스페이스에(`namespace2`)를 사용하여 `shareFromPVC` 소스 PVC를 지정하는 주석입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

Trident 다음을 읽습니다. `shareFromPVC` 대상 PVC에 주석을 추가하고 자체 저장 리소스가 없는 종속 볼륨으로 대상 PV를 생성하여 소스 PV를 가리키고 소스 PV 저장 리소스를 공유합니다. 대상 PVC와 PV는 정상적으로 바인딩된 것으로 나타납니다.

공유 볼륨 삭제

여러 네임스페이스에서 공유되는 볼륨을 삭제할 수 있습니다. Trident 소스 네임스페이스에서 볼륨에 대한 액세스를 제거하고 볼륨을 공유하는 다른 네임스페이스에 대한 액세스를 유지합니다. 볼륨을 참조하는 모든 네임스페이스가 제거되면 Trident 볼륨을 삭제합니다.

사용 `tridentctl get` 하위 볼륨을 쿼리하려면

를 사용하여[`tridentctl` 유틸리티를 실행하면 됩니다 `get` 하위 볼륨을 가져오는 명령입니다. 자세한 내용은 링크를 참조하세요:../trident-reference/tridentctl.html[`tridentctl` 명령 및 옵션].

```
Usage:
  tridentctl get [option]
```

플래그:

- `-h, --help`: 볼륨에 대한 도움말.
- `--parentOfSubordinate string`: 하위 소스 볼륨에 대한 쿼리를 제한합니다.
- `--subordinateOf string`: 볼륨의 하위 항목으로 쿼리를 제한합니다.

제한 사항

- Trident 대상 네임스페이스가 공유 볼륨에 쓰는 것을 막을 수 없습니다. 공유 볼륨 데이터를 덮어쓰는 것을 방지하려면 파일 잠금이나 다른 프로세스를 사용해야 합니다.
- 소스 PVC에 대한 액세스를 제거하여 취소할 수 없습니다. `shareToNamespace` 또는 `shareFromNamespace` 주석이나 삭제 `TridentVolumeReference` 크.알. 접근 권한을 취소하려면 하위 PVC를 삭제해야 합니다.
- 하위 볼륨에서는 스냅샷, 복제 및 미러링이 불가능합니다.

더 많은 정보를 원하시면

네임스페이스 간 볼륨 액세스에 대해 자세히 알아보려면 다음을 참조하세요.

- 방문하다"네임스페이스 간 볼륨 공유: 네임스페이스 간 볼륨 액세스를 만나보세요" .
- 데모를 시청하세요"넷앱TV" .

네임스페이스 전체에서 볼륨 복제

Trident 사용하면 동일한 Kubernetes 클러스터 내의 다른 네임스페이스에서 기존 볼륨이나 볼륨 스냅샷을 사용하여 새 볼륨을 만들 수 있습니다.

필수 조건

볼륨을 복제하기 전에 소스 및 대상 백엔드가 동일한 유형이고 동일한 스토리지 클래스를 가지고 있는지 확인하세요.

> ⓘ 네임스페이스 간 복제는 다음에 대해서만 지원됩니다. `ontap-san` 그리고 `ontap-nas` 스토리지 드라이버. 읽기 전용 복제본은 지원되지 않습니다.

빠른 시작

몇 단계만 거치면 볼륨 복제를 설정할 수 있습니다.

1. 볼륨을 복제하기 위해 소스 **PVC**를 구성합니다.

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.

2. 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여합니다.

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3. 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다.

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4. 대상 네임스페이스에 복제 **PVC**를 만듭니다.

대상 네임스페이스의 소유자는 소스 네임스페이스에서 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스 구성

보안을 보장하려면 네임스페이스 간에 볼륨을 복제하려면 소스 네임스페이스 소유자, 클러스터 관리자, 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계마다 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: PVC를 생성합니다.(`pvc1`) 소스 네임스페이스에서(`namespace1`) 대상 네임스페이스와 공유할 수 있는 권한을 부여합니다.(`namespace2`)를 사용하여 `cloneToNamespace` 주석.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident PV와 백엔드 스토리지 볼륨을 생성합니다.

- 쉼표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. 예를 들어, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` .
- 다음을 사용하여 모든 네임스페이스에 공유할 수 있습니다. `*` . 예를 들어, `trident.netapp.io/cloneToNamespace: *`
- PVC를 업데이트하여 다음을 포함할 수 있습니다. `cloneToNamespace` 언제든지 주석을 달 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에서 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하십시오.(`namespace2`).
3. 대상 네임스페이스 소유자: 소스 네임스페이스를 참조하는 대상 네임스페이스에 TridentVolumeReference CR을 생성합니다. `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. 대상 네임스페이스 소유자: PVC 생성(`pvc2`) 대상 네임스페이스에(`namespace2`)를 사용하여 `cloneFromPVC` 또는 `cloneFromSnapshot` , 그리고 `cloneFromNamespace` 소스 PVC를 지정하는 주석입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

제한 사항

- ontap-nas-economy 드라이버를 사용하여 프로비저닝된 PVC의 경우 읽기 전용 복제본은 지원되지 않습니다.

SnapMirror 사용하여 볼륨 복제

Trident 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 대상 볼륨 간의 미러 관계를 지원합니다. Trident Mirror Relationship(TMR)이라고 하는 네임스페이스가 지정된 사용자 지정 리소스 정의(CRD)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨(PVC) 간 미러 관계 생성
- 볼륨 간 미러 관계 제거
- 거울 속 관계를 깨세요
- 재해 상황(장애 조치) 동안 보조 볼륨을 홍보합니다.

- 계획된 장애 조치 또는 마이그레이션 중 클러스터 간에 애플리케이션의 손실 없는 전환을 수행합니다.

복제 전제 조건

시작하기 전에 다음 전제 조건이 충족되었는지 확인하세요.

ONTAP 클러스터

- * Trident *: ONTAP 백엔드로 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 Trident 버전 22.10 이상이 있어야 합니다.
- 라이선스: 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스는 소스 및 대상 ONTAP 클러스터 모두에서 활성화되어야 합니다. 참조하다 "ONTAP 의 SnapMirror 라이선싱 개요" 자세한 내용은.

  ONTAP 9.10.1부터 모든 라이선스는 여러 기능을 활성화하는 단일 파일인 NetApp 라이선스 파일(NLF)로 제공됩니다. 참조하다"ONTAP One에 포함된 라이센스" 자세한 내용은.

  SnapMirror 비동기 보호만 지원됩니다.

피어링

- 클러스터 및 **SVM**: ONTAP 스토리지 백엔드는 피어링되어야 합니다. 참조하다 "클러스터 및 SVM 피어링 개요" 자세한 내용은.

  두 ONTAP 클러스터 간 복제 관계에 사용된 SVM 이름이 고유한지 확인하세요.

- * Trident 및 SVM*: 피어링된 원격 SVM은 대상 클러스터의 Trident 에서 사용할 수 있어야 합니다.

지원되는 드라이버

NetApp Trident 다음 드라이버가 지원하는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 볼륨 복제를 지원합니다. **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` : FC** `ontap-san` : NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)

SnapMirror 사용한 볼륨 복제는 ASA r2 시스템에서는 지원되지 않습니다. ASA r2 시스템에 대한 정보는 다음을 참조하세요."ASA r2 스토리지 시스템에 대해 알아보세요" .

거울 **PVC** 만들기

다음 단계를 따르고 CRD 예제를 사용하여 기본 볼륨과 보조 볼륨 간의 미러 관계를 만듭니다.

단계

1. 기본 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. StorageClass 객체를 생성합니다. `trident.netapp.io/replication: true` 매개변수.

예

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. 이전에 생성한 StorageClass로 PVC를 생성합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. 지역 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident 볼륨의 내부 정보와 볼륨의 현재 데이터 보호(DP) 상태를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와서 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개변수를 사용하여 StorageClass를 생성합니다.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 목적지 및 소스 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident 구성된 관계 정책 이름(또는 ONTAP 의 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 보조(SnapMirror 대상) 역할을 하는 PVC를 생성합니다.

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident TridentMirrorRelationship CRD를 확인하고 해당 관계가 없으면 볼륨을 생성하지 못합니다. 관계가 존재하는 경우, Trident 새로운 FlexVol volume MirrorRelationship에 정의된 원격 SVM과 피어링된 SVM에 배치되도록 합니다.

볼륨 복제 상태

Trident 미러 관계(TMR)는 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 목적지 TMR에는 상태가 있는데, 이 상태는 Trident 원하는 상태가 무엇인지 알려줍니다. 목적지 TMR의 상태는 다음과 같습니다.

- 설립됨: 로컬 PVC는 미러 관계의 대상 볼륨이며, 이는 새로운 관계입니다.
- 홍보: 로컬 PVC는 읽기/쓰기가 가능하고 마운트 가능하며, 현재 미러 관계는 적용되지 않습니다.
- 재설정: 로컬 PVC가 미러 관계의 대상 볼륨이며 이전에도 해당 미러 관계에 있었습니다.

- 대상 볼륨이 소스 볼륨과 관계를 맺은 적이 있는 경우에는 재설정된 상태를 사용해야 합니다. 재설정된 상태는 대상 볼륨의 내용을 덮어쓰기 때문입니다.
- 볼륨이 이전에 소스와 관계가 없었던 경우 재설정된 상태는 실패합니다.

계획되지 않은 장애 조치 중 **2**차 **PVC** 홍보

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 *spec.state* 필드를 다음으로 업데이트합니다. `promoted` .

계획된 장애 조치 중 보조 **PVC** 홍보

계획된 장애 조치(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격합니다.

단계

1. 기본 Kubernetes 클러스터에서 PVC의 스냅샷을 만들고 스냅샷이 생성될 때까지 기다립니다.
2. 기본 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 얻습니다.

   예

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship* CR의 *spec.state* 필드를 _promoted_로 업데이트하고 _spec.promotedSnapshotHandle_을 스냅샷의 internalName으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 상태(status.state 필드)가 승격되었는지 확인합니다.

장애 조치 후 미러 관계 복원

미러 관계를 복원하기 전에 새로운 기본 관계로 만들려는 측면을 선택하세요.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.remoteVolumeHandle* 필드 값이 업데이트되었는지 확인하세요.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.mirror* 필드를 다음으로 업데이트합니다. `reestablished` .

추가 작업

Trident 기본 및 보조 볼륨에서 다음 작업을 지원합니다.

1차 PVC를 새로운 2차 PVC로 복제합니다.

기본 PVC와 보조 PVC가 이미 있는지 확인하세요.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 기본(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새로운 2차(대상) PVC에 대한 기본(소스) 클러스터에 새로운 TridentMirrorRelationship CRD를 만듭니다.

미러링된 1차 또는 2차 PVC 크기 조정

PVC는 일반적으로 크기를 조절할 수 있으며, 데이터 양이 현재 크기를 초과하면 ONTAP 모든 대상 flevxols를 자동으로 확장합니다.

PVC에서 복제 제거

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행합니다.

- 2차 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promoted_로 업데이트합니다.

PVC(이전에 미러링됨) 삭제

Trident 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR 삭제

미러링된 관계의 한쪽에서 TMR을 삭제하면 Trident 삭제를 완료하기 전에 나머지 TMR이 승격 상태로 전환됩니다. 삭제를 위해 선택된 TMR이 이미 *promoted* 상태인 경우 기존 미러 관계가 없으므로 TMR이 제거되고 Trident 로컬 PVC를 *ReadWrite*_로 승격합니다. 이 삭제로 *ONTAP* 의 로컬 볼륨에 대한 *SnapMirror* 메타데이터가 해제됩니다. 이 볼륨이 나중에 미러 관계에서 사용되는 경우 새 미러 관계를 만들 때 _설정된 볼륨 복제 상태를 가진 새 TMR을 사용해야 합니다.

ONTAP 이 온라인일 때 미러 관계 업데이트

미러 관계는 설정된 후 언제든지 업데이트할 수 있습니다. 당신은 사용할 수 있습니다 `state: promoted` 또는 `state: reestablished` 관계를 업데이트하는 필드입니다. 대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복원할 특정 스냅샷을 지정할 수 있습니다.

ONTAP 이 오프라인일 때 미러 관계 업데이트

Trident ONTAP 클러스터에 직접 연결되지 않고도 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. TridentActionMirrorUpdate의 다음 예제 형식을 참조하세요.

예

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state`TridentActionMirrorUpdate CRD의 상태를 반영합니다. *Succeeded*, *In Progress*, 또는 *Failed* 값을 가질 수 있습니다.

CSI 토폴로지 사용

Trident Kubernetes 클러스터에 있는 노드에 볼륨을 선택적으로 생성하고 첨부할 수 있습니다. "CSI 토폴로지 기능" .

개요

CSI 토폴로지 기능을 사용하면 지역 및 가용성 영역을 기반으로 볼륨에 대한 액세스를 노드 하위 집합으로 제한할 수 있습니다. 오늘날 클라우드 공급업체는 Kubernetes 관리자가 영역 기반 노드를 생성할 수 있도록 지원합니다. 노드는 한 지역 내의 여러 가용성 영역에 위치할 수도 있고, 여러 지역에 걸쳐 위치할 수도 있습니다. 다중 구역 아키텍처의 워크로드에 대한 볼륨 프로비저닝을 용이하게 하기 위해 Trident CSI 토폴로지를 사용합니다.

CSI 토폴로지 기능에 대해 자세히 알아보세요 "여기" .

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- 와 함께 `VolumeBindingMode` 로 설정 `Immediate` Trident 토폴로지를 인식하지 않고 볼륨을 생성합니다. 볼륨 바인딩과 동적 프로비저닝은 PVC가 생성될 때 처리됩니다. 이것은 기본값입니다 `VolumeBindingMode` 토폴로지 제약을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청하는 포드의 스케줄링 요구 사항에 대한 종속성 없이 생성됩니다.
- 와 함께 `VolumeBindingMode` 로 설정 `WaitForFirstConsumer` PVC에 대한 영구 볼륨의 생성 및 바인딩은 PVC를 사용하는 포드가 예약되고 생성될 때까지 지연됩니다. 이런 방식으로 토폴로지 요구 사항에 의해 적용되는 스케줄링 제약 조건을 충족하는 볼륨이 생성됩니다.

그만큼 `WaitForFirstConsumer` 바인딩 모드에는 토폴로지 레이블이 필요하지 않습니다. 이 기능은 CSI 토폴로지 기능과 별도로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 활용하려면 다음이 필요합니다.

- Kubernetes 클러스터를 실행 중"지원되는 Kubernetes 버전"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 클러스터의 노드에는 토폴로지 인식을 소개하는 레이블이 있어야 합니다 .(`topology.kubernetes.io/region` 그리고 `topology.kubernetes.io/zone`). Trident 가 토폴로지를 인식할 수 있도록 Trident 설치하기 전에 이러한 레이블이 클러스터의 노드에 있어야 합니다.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

1단계: 토폴로지 인식 백엔드 만들기

Trident 스토리지 백엔드는 가용성 영역에 따라 볼륨을 선택적으로 프로비저닝하도록 설계될 수 있습니다. 각 백엔드는 선택 사항을 수행할 수 있습니다. `supportedTopologies` 지원되는 영역 및 지역 목록을 나타내는 블록입니다. 이러한 백엔드를 사용하는 StorageClass의 경우, 지원되는 지역/영역에서 예약된 애플리케이션에서 요청하는 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

`supportedTopologies`백엔드별 지역 및 영역 목록을 제공하는 데 사용됩니다. 이러한 지역과 영역은 StorageClass에서 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공되는 지역 및 영역의 하위 집합을 포함하는 StorageClass의 경우 Trident 백엔드에 볼륨을 생성합니다.

정의할 수 있습니다 `supportedTopologies` 저장 풀당도 마찬가지입니다. 다음 예를 참조하세요.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

이 예에서는 `region` 그리고 `zone` 라벨은 저장 풀의 위치를 나타냅니다. `topology.kubernetes.io/region` 그리고 `topology.kubernetes.io/zone` 저장 풀을 어디에서 사용할 수 있는지 지정합니다.

2단계: 토폴로지를 인식하는 **StorageClass** 정의

클러스터의 노드에 제공된 토폴로지 레이블을 기반으로 토폴로지 정보를 포함하도록 StorageClass를 정의할 수 있습니다. 이를 통해 PVC 요청에 대한 후보 역할을 하는 스토리지 풀과 Trident 에서 프로비저닝한 볼륨을 활용할 수 있는 노드 하위 집합이 결정됩니다.

다음 예를 참조하세요.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - key: topology.kubernetes.io/zone
    values:
      - us-east1-a
      - us-east1-b
  - key: topology.kubernetes.io/region
    values:
      - us-east1
parameters:
  fsType: ext4
```

위에 제공된 StorageClass 정의에서 `volumeBindingMode` 로 설정됩니다 `WaitForFirstConsumer` . 이 StorageClass로 요청된 PVC는 Pod에서 참조될 때까지 작업이 수행되지 않습니다. 그리고, `allowedTopologies` 사용할 구역과 지역을 제공합니다. 그만큼 `netapp-san-us-east1` StorageClass는 PVC를 생성합니다. `san-backend-us-east1` 백엔드는 위에 정의되어 있습니다.

3단계: **PVC** 만들기 및 사용

StorageClass를 생성하고 백엔드에 매핑했으므로 이제 PVC를 생성할 수 있습니다.

예를 보세요 `spec` 아래에:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

이 매니페스트를 사용하여 PVC를 생성하면 다음과 같은 결과가 발생합니다.

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Trident 볼륨을 생성하고 이를 PVC에 연결하려면 포드에서 PVC를 사용하세요. 다음 예를 참조하세요.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

이 podSpec은 Kubernetes에 현재 존재하는 노드에서 Pod를 예약하도록 지시합니다. `us-east1` 지역 및 해당 지역에 있는 모든 노드 중에서 선택하십시오. `us-east1-a` 또는 `us-east1-b` 구역.

다음 출력을 확인하세요.

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

백엔드를 업데이트하여 포함합니다. `supportedTopologies`

기존 백엔드를 업데이트하여 다음 목록을 포함할 수 있습니다. `supportedTopologies` 사용 중 `tridentctl backend update` . 이는 이미 프로비저닝된 볼륨에는 영향을 미치지 않으며, 후속 PVC에만 사용됩니다.

더 많은 정보를 찾아보세요

- "컨테이너 리소스 관리"
- "노드 선택기"
- "친화성과 반친화성"
- "오염과 관용"

스냅샷 작업

영구 볼륨(PV)의 Kubernetes 볼륨 스냅샷을 사용하면 볼륨의 특정 시점 복사본을 만들 수 있습니다. Trident 사용하여 생성된 볼륨의 스냅샷을 생성하고, Trident 외부에서 생성된 스냅샷을 가져오고, 기존 스냅샷에서 새 볼륨을 생성하고, 스냅샷에서 볼륨 데이터를 복구할 수 있습니다.

개요

볼륨 스냅샷은 다음에서 지원됩니다. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `gcp-cvs` , `azure-netapp-files` , 그리고 `google-cloud-netapp-volumes` 운전자.

시작하기 전에

스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 사용자 정의 리소스 정의(CRD)가 필요합니다. 이는 Kubernetes 오케스트레이터(예: Kubeadm, GKE, OpenShift)의 책임입니다.

Kubernetes 배포판에 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요.볼륨 스냅샷 컨트롤러 배포 .

> GKE 환경에서 주문형 볼륨 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마세요. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

볼륨 스냅샷 생성

단계

1. 생성하다 `VolumeSnapshotClass` 자세한 내용은 다음을 참조하세요."볼륨 스냅샷 클래스" .
   - 그만큼 `driver` Trident CSI 드라이버를 가리킨다.
   - `deletionPolicy`될 수 있다 `Delete` 또는 `Retain` . 설정 시 `Retain` , 스토리지 클러스터의 기본 물리적 스냅샷은 다음과 같은 경우에도 유지됩니다. `VolumeSnapshot` 객체가 삭제되었습니다.

     예

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. 기존 PVC의 스냅샷을 만듭니다.

   예시
   - 이 예제에서는 기존 PVC의 스냅샷을 만듭니다.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - 이 예제에서는 PVC에 대한 볼륨 스냅샷 객체를 생성합니다. `pvc1` 그리고 스냅샷의 이름은 다음과 같이 설정됩니다. `pvc1-snap` . VolumeSnapshot은 PVC와 유사하며 다음과 연관됩니다. `VolumeSnapshotContent` 실제 스냅샷을 나타내는 객체입니다.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- 당신은 식별할 수 있습니다 `VolumeSnapshotContent` ~에 대한 객체 `pvc1-snap` VolumeSnapshot을 설명하여 보세요. 그만큼 `Snapshot Content Name` 이 스냅샷을 제공하는 VolumeSnapshotContent 객체를 식별합니다. 그만큼 `Ready To Use` 매개변수는 스냅샷을 사용하여 새로운 PVC를 생성할 수 있음을 나타냅니다.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

볼륨 스냅샷에서 **PVC** 생성

사용할 수 있습니다 `dataSource` VolumeSnapshot이라는 이름을 사용하여 PVC를 생성하려면 `<pvc-name>` 데이터의 출처로서. PVC를 만든 후에는 포드에 부착하여 다른 PVC와 마찬가지로 사용할 수 있습니다.

> PVC는 소스 볼륨과 동일한 백엔드에 생성됩니다. 참조하다"KB: Trident PVC 스냅샷에서 PVC를 생성하는 작업은 대체 백엔드에서 생성할 수 없습니다." .

다음 예제에서는 다음을 사용하여 PVC를 생성합니다. `pvc1-snap` 데이터 소스로.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

볼륨 스냅샷 가져오기

Trident 다음을 지원합니다."Kubernetes 사전 프로비저닝 스냅샷 프로세스" 클러스터 관리자가 다음을 생성할 수 있도록 합니다. `VolumeSnapshotContent` Trident 외부에서 생성된 객체 및 가져오기 스냅샷.

시작하기 전에

Trident 스냅샷의 부모 볼륨을 생성하거나 가져와야 합니다.

단계

1. 클러스터 관리자: 생성 `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 개체입니다. 이렇게 하면 Trident 에서 스냅샷 워크플로가 시작됩니다.
   - 백엔드 스냅샷의 이름을 지정하세요. `annotations` ~처럼 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name"> .`
   - 지정하다 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` ~에 `snapshotHandle` . 이것은 외부 스냅샷터가 Trident 에 제공하는 유일한 정보입니다. `ListSnapshots` 부르다.

     그만큼 `<volumeSnapshotContentName>` CR 명명 제약으로 인해 백엔드 스냅샷 이름과 항상 일치할 수는 없습니다.

     예

     다음 예제에서는 다음을 생성합니다. `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 객체 `snap-01` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. 클러스터 관리자: 생성 `VolumeSnapshot` 참조하는 CR `VolumeSnapshotContent` 물체. 이것은 사용에 대한 액세스를 요청합니다. `VolumeSnapshot` 주어진 네임스페이스에서.

   예

   다음 예제에서는 다음을 생성합니다. `VolumeSnapshot` CR이 명명됨 `import-snap` 참조하는 `VolumeSnapshotContent` 명명된 `import-snap-content` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. 내부 처리(작업 필요 없음): 외부 스냅샷터는 새로 생성된 것을 인식합니다. `VolumeSnapshotContent` 그리고 실행됩니다 `ListSnapshots` 부르다. Trident 다음을 생성합니다. `TridentSnapshot` .
   - 외부 스냅샷터는 다음을 설정합니다. `VolumeSnapshotContent` 에게 `readyToUse` 그리고 `VolumeSnapshot` 에게 `true` .
   - Trident 돌아온다 `readyToUse=true` .
4. 모든 사용자: 만들기 `PersistentVolumeClaim` 새로운 것을 참조하기 위해 `VolumeSnapshot` , 여기서 `spec.dataSource` (또는 `spec.dataSourceRef`) 이름은 `VolumeSnapshot` 이름.

예

다음 예제에서는 다음을 참조하는 PVC를 생성합니다. `VolumeSnapshot` 명명된 `import-snap` .

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

스냅샷을 사용하여 볼륨 데이터 복구

스냅샷 디렉토리는 기본적으로 숨겨져 있어 볼륨 프로비저닝의 최대 호환성을 용이하게 합니다. `ontap-nas` 그리고 `ontap-nas-economy` 운전자. 활성화 `.snapshot` 스냅샷에서 직접 데이터를 복구할 수 있는 디렉토리입니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 적용된 변경 사항은 손실됩니다.

스냅샷에서 제자리 볼륨 복원

Trident 다음을 사용하여 스냅샷에서 빠르고 정확한 볼륨 복원을 제공합니다. `TridentActionSnapshotRestore` (TASR) CR. 이 CR은 필수 Kubernetes 작업으로 작동하며 작업이 완료된 후에는 유지되지 않습니다.

Trident 스냅샷 복원을 지원합니다. `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` , `google-cloud-netapp-volumes` , 그리고 `solidfire-san` 운전자.

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인하세요.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인하세요.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예제에서는 PVC에 대한 CR을 생성합니다. `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot` .

   TASR CR은 PVC 및 VS가 있는 네임스페이스에 있어야 합니다.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 스냅샷에서 복원하려면 CR을 적용합니다. 이 예제는 스냅샷에서 복원합니다. `pvc1` .

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

결과

Trident 스냅샷에서 데이터를 복원합니다. 스냅샷 복원 상태를 확인할 수 있습니다.

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 대부분의 경우, Trident 실패 시 자동으로 작업을 다시 시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 액세스 권한이 없는 Kubernetes 사용자는 애플리케이션 네임스페이스에 TASR CR을 생성하기 위해 관리자로부터 권한을 부여받아야 할 수도 있습니다.

연관된 스냅샷이 있는 **PV** 삭제

연관된 스냅샷이 있는 영구 볼륨을 삭제하면 해당 Trident 볼륨이 "삭제 중 상태"로 업데이트됩니다. 볼륨 스냅샷을 제거하여 Trident 볼륨을 삭제합니다.

볼륨 스냅샷 컨트롤러 배포

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않으면 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

필요한 경우 열어주세요 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 그리고 업데이트 `namespace` 네임스페이스에.

관련 링크

- "볼륨 스냅샷"
- "볼륨 스냅샷 클래스"

볼륨 그룹 스냅샷 작업

영구 볼륨(PV)의 Kubernetes 볼륨 그룹 스냅샷 NetApp Trident 여러 볼륨(볼륨 스냅샷 그룹)의 스냅샷을 생성하는 기능을 제공합니다. 이 볼륨 그룹 스냅샷은 동일한 시점에 생성된 여러 볼륨의 복사본을 나타냅니다.

VolumeGroupSnapshot은 베타 API가 포함된 Kubernetes의 베타 기능입니다. VolumeGroupSnapshot에 필요한 최소 버전은 Kubernetes 1.32입니다.

볼륨 그룹 스냅샷 생성

볼륨 그룹 스냅샷은 다음과 같이 지원됩니다. `ontap-san` 드라이버는 iSCSI 프로토콜에만 적용되며, 아직 Fibre Channel(FCP)이나 NVMe/TCP에서는 지원되지 않습니다. .시작하기 전에

- Kubernetes 버전이 K8s 1.32 이상인지 확인하세요.
- 스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 사용자 정의 리소스 정의(CRD)가 필요합니다. 이는 Kubernetes 오케스트레이터(예: Kubeadm, GKE, OpenShift)의 책임입니다.

  Kubernetes 배포판에 외부 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요.볼륨 스냅샷 컨트롤러 배포 .

  > GKE 환경에서 주문형 볼륨 그룹 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마세요. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

- 스냅샷 컨트롤러 YAML에서 다음을 설정합니다. `CSIVolumeGroupSnapshot` 볼륨 그룹 스냅샷이 활성화되도록 기능 게이트를 'true'로 설정합니다.
- 볼륨 그룹 스냅샷을 생성하기 전에 필요한 볼륨 그룹 스냅샷 클래스를 생성하세요.
- VolumeGroupSnapshot을 생성하려면 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하세요.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하세요. 자세한 내용은 다음을 참조하세요."볼륨 그룹 스냅샷 클래스" .

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 기존 저장 클래스를 사용하여 필요한 라벨이 있는 PVC를 만들거나, 이러한 라벨을 기존 PVC에 추가합니다.

  다음 예제에서는 다음을 사용하여 PVC를 생성합니다. `pvc1-group-snap` 데이터 소스 및 레이블로 `consistentGroupSnapshot: groupA` . 요구 사항에 따라 레이블 키와 값을 정의합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- 동일한 레이블로 VolumeGroupSnapshot을 만듭니다.(`consistentGroupSnapshot: groupA`) PVC에 지정됨.

  이 예제에서는 볼륨 그룹 스냅샷을 생성합니다.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

그룹 스냅샷을 사용하여 볼륨 데이터 복구

볼륨 그룹 스냅샷의 일부로 생성된 개별 스냅샷을 사용하여 개별 영구 볼륨을 복원할 수 있습니다. 볼륨 그룹 스냅샷을 단위로 복구할 수 없습니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 적용된 변경 사항은 손실됩니다.

스냅샷에서 제자리 볼륨 복원

Trident 다음을 사용하여 스냅샷에서 빠르고 정확한 볼륨 복원을 제공합니다. `TridentActionSnapshotRestore` (TASR) CR. 이 CR은 필수 Kubernetes 작업으로 작동하며 작업이 완료된 후에는 유지되지 않습니다.

자세한 내용은 다음을 참조하세요. "스냅샷에서 제자리 볼륨 복원".

연관된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- 그룹의 개별 스냅샷이 아닌 VolumeGroupSnapshots 전체를 삭제할 수 있습니다.
- 스냅샷이 있는 동안 PersistentVolume이 삭제되면 Trident 해당 볼륨을 "삭제 중" 상태로 전환합니다. 볼륨을 안전하게 제거하기 전에 스냅샷을 제거해야 하기 때문입니다.
- 그룹화된 스냅샷을 사용하여 복제본을 만든 다음 그룹을 삭제하려는 경우 복제본 분할 작업이 시작되고 분할이 완료될 때까지 그룹을 삭제할 수 없습니다.

볼륨 스냅샷 컨트롤러 배포

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않으면 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

필요한 경우 열어주세요 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 그리고 업데이트 `namespace` 네임스페이스에.

관련 링크

- "볼륨 그룹 스냅샷 클래스"
- "볼륨 스냅샷"